

开发专家之 Sun ONE

# JSP 应用开发详解

飞思科技产品研发中心 编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

本书基于完整的 J2EE 框架，对 JSP 编程技术进行了深入而全面的介绍。全书分为 6 篇共 22 章，包括 JSP 应用开发基础、JSP 核心语法及实例解析、Servlet 技术在 JSP 开发中的应用、JDBC、基于 XML 的 JSP 应用以及 JSP 的完整网站开发实例。本书基于完整的 J2EE 框架，对 JSP 编程技术进行了深入而全面的介绍，重点在于 JSP 和其他技术的综合应用开发。书中的例程对实际的应用开发有非常强的借鉴意义。

本书适合于专业和准专业的 Java 程序员阅读，也可作为正在进行 Java 开发的各类程序员的必备 Java 参考书。

未经许可，不得以任何方式复制或抄袭本书的部分或全部内容。  
版权所有，翻版必究。

图书在版编目 (CIP) 数据

JSP 应用开发详解/飞思科技产品研发中心编著. —北京：电子工业出版社，2002.1

(开发专家之 Sun ONE)

ISBN 7-5053-7365-x

I. J... II. 飞... III. Java 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2001) 第 093134 号

责任编辑：郭 晶 刘韦韦

印 刷：北京天宇星印刷厂

出版发行：电子工业出版社 <http://www.phei.com.cn>

北京海淀区万寿路 173 信箱 邮编：100036

经 销：各地新华书店

开 本：787×1092 1/16 印张：33 字数：844.8 千字 附光盘 1 张

版 次：2002 年 1 月第 1 版 2002 年 1 月第 1 次印刷

印 数：4000 册 定价：49.00 元 (含光盘)

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：(010) 68279077

## 出版说明

“开发专家”是电子工业出版社计算机图书研发部长期以来精心培育的计算机科学技术类本版品牌。这个品牌是由多个专题系列组成的横向大系列，涵盖了计算机技术的各个方面，特别是一直受到极大关注的程序开发类系列，例如《开发专家之数据库》、《开发专家之网络编程》、《开发专家之 Delphi》以及《开发专家之 Sun ONE》等。这些专题系列基于各自的角度，从纵向上包含了该专题的所有内容。因此，整个“开发专家”的品牌架构纵横交错，囊括了所有的计算机技术和所有的技术层面，海纳百川而又极具可扩展性。

“开发专家”的作者队伍主要依托于“飞思科技产品研发中心”。“飞思科技产品研发中心”由专业的策划人员、权威的技术专家和资深的作者队伍共同构成。在图书的出版上，形成了以研发为基础、以出版为中心、以服务为支持的专业化出版框架和流程。通过深入的市场调查和技术跟踪，在综合了技术需求和读者焦点等因素的基础上，形成各系列丛书的写作重点和大纲，然后聘请业界的最前沿学者进行写作。同时，策划工作全程介入写作进程，严格控制写作质量，用最专业的技术背景、最深刻的理论基础、最具代表性的案例、最能为专业读者接受的形式，为读者提供品质最佳的图书产品，体现了出版者和著作者的完美结合。

多年来，计算机图书研发部始终把创造社会效益摆在首位，秉承一切为国内计算机技术专业读者服务的精神，为推动国内 IT 技术发展、为体现国内技术的原创水平，穷尽所有的创意与努力，将出版者的命运与读者的支持紧紧地连在了一起。

在此，我们临出版之残酷竞争而不惧，旌旗猎猎而异军突起，这与广大读者的支持是分不开的。为使我们的脚步更坚实、使我们的队伍永远保持活力和创造力，我们期待着您能为我们的前进贡献出您的意见和建议。同时，我们也在等待着您的加入。

我们的联系方式：

电 话： （010）68134545

E-mail： support@fecit.com.cn

网 址： <http://www.fecit.com.cn> <http://www.fecit.net>

电子工业出版社计算机图书研发部

# 前 言

## 关于本套丛书

从来没有任何事物像互联网那样，对人类的活动产生如此深刻的影响，无论是政府、企业，以及个人，莫不如此。与此同时，IT 工业也正面临着—场变革——传统应用向基于 Internet/Web 的服务模式转化。

翻开历史，我们可以看到互联网的形成和发展就是以分布性、开放性和平台无关性为基础，这是 Internet 与生俱有的属性。随着互联网应用的发展，又引入了诸如 RPC/COM/CORBA 等技术，但这些的技术在实际应用中，又存在着很多不足和局限。它们的特定协议也难以通过防火墙，因而不适于 Web 上的应用开发。为了进一步开发基于 Web 的应用，相继出现了 Sun 公司的 Sun ONE (Open Net Environment 开放网络环境) 和 Microsoft 公司的 .NET 两大 Web 服务技术体系。其中，Sun ONE 以 Java 技术为核心，更接近或者满足于互联网在智能化 Web 服务上对分布性、开放性和平台无关性的要求，同时其在健壮性、安全性、组件化等方面也更为成熟稳定，获得了众多 IT 厂商和产品的支持，是目前惟一在市场上得到了广泛应用的技术体系。

Sun ONE 体系结构以 Java 语言为核心，包括 J2SE/J2EE/J2ME，并基于一系列开放和流行标准、技术及协议。要特别指出的是，Sun ONE 体系结构本身作为开放式体系结构，在得到 IBM/BEA/Oracle/Sybase 等这些 IT 巨擎支持的同时，更得到了互联网上 Open Source 社区的青睐。我们很容易地从网上免费获得和使用包括 Java 集成开发环境、Java 数据库，甚至是中间件 (Application Server) 服务器等产品，以及它们的源代码。这对于加速国内中小企业的信息化建设和自有知识产权产品开发、提高企业应用和软件行业的整体水平，无疑是一次难得的机会。

综观国内的技术发展，广大的 Java 程序开发人员以及正在转向 Java 体系进行开发的技术人员虽然已面临这一令人激动和鼓舞的转型期，却苦于没有足够的相关资料和文献，尤其对国内的最新 Java 技术动态和技术现状知之甚少，而图书市场上 Java 的书籍尽管汗牛充栋，但精品罕见，能反映出 J2EE 以及 Sun ONE 的框架全貌的书籍更是奇缺。

电子工业出版社计算机图书研发部为进一步推动国内 Java 技术的应用与发展，不失时机地推出了《开发专家之 Sun ONE》系列丛书。

本套丛书以 Sun ONE 整体架构为基础，全面体现了 Sun ONE 的技术核心——Java 的应用开发。丛书从各个角度深入 Java 应用开发的各个层面，涵盖了 Java 技术的所有重要思想和实践，体现了最新的 Java 技术进展和动态，大幅度提升读者的理论和应用水平。同时，丛书重点突出实用性。书中引入了大量的行业应用范例，使读者不仅能快速掌握开发技能，而且对于开发者进行综合系统分析也有所裨益。



## 关于本书

JSP 已经成为开发 Web 动态网站的重要而快速、有效的工具，它是全新的网络服务器端编程环境。JSP 充分利用了 Java 的强大功能，是一种优秀的服务器端技术。随着 Java 技术的日益成熟和流行，JSP 技术在网络编程中也变得越来越重要。由于 JSP 基于强大的 Java 语言，具有极强的扩展能力，良好的收缩性，以及与平台无关的开发特性，在根据 Java 平台构建动态商务网站成为主流的今天，JSP 有着其他技术所不具备的优势。

本书基于完整的 J2EE 框架，对 JSP 编程技术进行了深入而全面的介绍，重点在于 JSP 和其他技术的综合应用开发。书中详细介绍了 JSP、Servlet、JDBC、XML 等强大而又先进的动态网站开发技术，并结合大量开发实例，剖析了如何基于 Java 平台快速构建高速、高效的电子商务平台，并将 JSP 与 ASP、PHP 等传统技术做了比较，充分展示了 JSP 的强大功能和用 JSP 开发网站的优越性。全书分为 6 篇共 22 章，包括 JSP 应用开发基础、JSP 核心语法及实例解析、Servlet 技术在 JSP 开发中的应用、JDBC、基于 XML 的 JSP 应用以及 JSP 的完整网站开发实例。书中的例程对实际的应用开发有非常强的借鉴意义。

本书适合于专业和准专业的 Java 程序员阅读，也可作为正在进行 Java 开发的各类程序员的必备 Java 参考书。

本书由飞思科技产品研发中心策划并组织编写，王夕宁、莫迪、马文飞、张箐、黎昌杰、胡韬、吴瑞鹏、李景彬、李艳燕、王红安等人参加了本书的写作工作。同时，在本书的写作过程中得到了董毅先生的大力支持和协助，他提出了大量的参考性意见使本书增色不少，在此表示衷心的感谢。

当然，限于作者水平，加之时间仓促，书中不足之处难免，敬请读者批评指正。

我们的联系方式：

电 话： (010) 68134545 68134811

E-mail: support@fecit.com.cn

网 址: <http://www.fecit.com.cn> <http://www.fecit.net>

飞思科技产品研发中心

# 目 录

## 第一篇 JSP 应用开发基础篇

第 1 章 JSP 技术概论 .....	3
1.1 动态网站技术介绍 .....	3
1.2 JSP 技术原理 .....	4
1.2.1 JSP 语言及其特点 .....	4
1.2.2 JSP 实现原理 .....	6
1.3 JSP 与其他技术的比较 .....	9
1.4 JSP 应用开发的未来 .....	14
1.5 JSP 实例讲解 .....	15
第 2 章 JSP 在应用开发中的语言结构 .....	17
2.1 超文本标记语言 .....	17
2.1.1 超文本标记语言语法 .....	17
2.2.2 动态超文本标记语言 .....	19
2.2 JSP 开发中的 Java 应用 .....	21
2.2.1 在 JSP 开发中的 Java 语法 .....	21
2.2.2 JSP 开发中的 Java 应用 .....	28
第 3 章 JSP 运行环境的安装配置 .....	35
3.1 JSP 对运行环境的要求 .....	35
3.1.1 对硬件条件的要求 .....	35
3.1.2 对操作系统的要求 .....	35
3.1.3 对软件环境的要求 .....	36
3.2 JSP 开发和运行环境的安装配置 .....	37
3.2.1 JDK 的安装配置 .....	37
3.2.2 应用程序服务器的安装配置 .....	38
3.2.3 Java 集成开发环境的安装配置 .....	44
3.2.4 数据库的安装配置 .....	48
3.3 Tomcat UNIX/Windows 服务器安装与配置 .....	53
3.3.1 Tomcat 的安装 .....	53
3.3.2 Tomcat 的配置 .....	54
3.3.3 Tomcat 和 Web Server 的配合 .....	58
3.3.4 在 Tomcat 中建立新的 Web 应用程序 .....	59
3.4 安装配置中的常见问题与解决方法 .....	62

## 第二篇 JSP 核心内容

第 4 章 JSP 语法详解 .....	67
4.1 JSP 程序的基本结构 .....	67
4.2 JSP 指令 .....	69

4.2.1	Page 指令 .....	70
4.2.2	Include 指令 .....	72
4.3	JSP 动作指令 .....	76
4.3.1	Include 指令 .....	76
4.3.2	Forward 指令 .....	78
4.3.3	UseBean 指令 .....	80
4.3.4	GetProperty 指令 .....	81
4.3.5	SetProperty 指令 .....	82
4.3.6	Plugin 指令 .....	84
4.4	JSP 语法实例 NowTime .....	92
第 5 章	JSP 的内建对象及实例分析 .....	99
5.1	JSP 的内建对象概述 .....	99
5.1.1	JSP 的内建对象一: request .....	100
5.1.2	JSP 的内建对象二: response .....	111
5.1.3	JSP 的内建对象三: out .....	113
5.1.4	JSP 的内建对象四: Session .....	118
5.1.5	JSP 的内建对象五: pageContext .....	120
5.1.6	JSP 的内建对象六: application .....	121
5.1.7	JSP 的内建对象七: config .....	124
5.1.8	JSP 的内建对象八: page .....	124
5.2	JSP 的内建对象实例 (简单的购物车) .....	124
第 6 章	JavaBeans 在 JSP 中的应用 .....	137
6.1	JavaBeans 的概念 .....	137
6.2	JavaBeans 技术概述 .....	138
6.2.1	JavaBeans 的属性 .....	138
6.2.2	JavaBeans 的方法 .....	142
6.3	JavaBeans 的应用 .....	142
6.3.1	在 JSP 中使用 JavaBeans .....	142
6.3.2	购物车范例 .....	144
6.3.3	多选框范例 .....	147
6.3.4	猜数字范例 .....	149
6.3.5	封装数据库范例 .....	152
第 7 章	JSP 应用中的文件操作 .....	157
7.1	从文件中读取数据 .....	157
7.2	向文件中写入数据 .....	163
7.3	向文件中追加数据 .....	167
7.4	构建计数器实例 .....	170

### 第三篇 Servlet 技术的应用

第 8 章	Servlet 技术	177
8.1	Servlet 技术概述	177
8.1.1	什么是 Servlet	177
8.1.2	Servlet 与其他开发技术 (CGI—BIN) 的比较	178
8.2	Servlet 与 JSP 之间的关系	180
8.3	Servlet 应用范围及其缺陷	184
8.4	Servlet 的生命周期	185
8.5	Servlet 常用类接口	188
8.5.1	HttpServlet	188
8.5.2	HttpServletRequest	192
8.5.3	HttpServletResponse	194
8.5.4	HttpSession	197
8.5.5	ServletConfig	199
8.5.6	ServletContext	199
8.6	JSP 内建对象与 Servlet 中类的对应关系	199
第 9 章	JSP 与 Servlet 结合编程	209
9.1	JSP 技术网站开发的两种模式	209
9.1.1	模式一: JSP+JavaBeans	209
9.1.2	模式二: JSP+Servlet+JavaBeans	209
9.2	两种模式的实例	210
9.2.1	使用模式一实现	218
9.2.2	使用模式二实现	252
9.3	JSP 发信的工作原理	272
9.4	JSP 在发信中的应用	274

### 第四篇 在 JSP 开发中使用数据库

第 10 章	JSP 开发中的数据库工作原理	281
10.1	数据库及 SQL 语句	281
10.1.1	建表、修改和删除表的语句	281
10.1.2	查询语句	282
10.1.3	插入、更新和删除语句	282
10.1.4	条件子句	282
10.1.5	Oracle 的函数	283
10.2	JDBC 技术工作原理	284
10.3	JDBC 四种类型的驱动	286
10.3.1	JDBC-ODBC Bridge	286
10.3.2	JDBC Native Bridge	286
10.3.3	JDBC-Network Bridge	287

10.3.4	Pure Java JDBC Drive.....	287
10.4	JDBC 接口 .....	287
10.4.1	Driver .....	287
10.4.2	DriverManager .....	287
10.4.3	Connection .....	290
10.4.4	Statement.....	291
10.4.5	ResultSet .....	294
10.5	数据库连接 JDBC 实例 .....	296
10.5.1	建立与 Oracle 的连接 .....	296
10.5.2	建立与 My SQL 的连接.....	297
10.5.3	实例.....	299
第 11 章	在 JSP 和 Servlet 中使用 JDBC 技术 .....	321
11.1	在 JSP 中使用 JDBC 技术 .....	321
11.1.1	在 JSP 页面中直接连接数据库 .....	321
11.1.2	使用 JavaBeans 技术封装对数据库的操作 .....	327
11.2	在 Servlet 中使用 JDBC 技术.....	331
第 12 章	数据库的 Connection Pool 技术 .....	345
12.1	什么是 Connection Pool 技术 .....	345
12.2	Connection Pool 技术的优点 .....	346
12.3	Connection Pool 技术的原理 .....	346
12.3.1	ConnectionPool 类 .....	347
12.3.2	PoolManager 类 .....	349
12.4	常用服务器的 Connection Pool 的配置工作 .....	351
12.4.1	weblogic 服务器的配置 .....	351
12.4.2	resin 服务器的配置 .....	352
12.5	Connection Pool 使用实例 .....	352
	<b>第五篇 基于 XML 的 JSP 应用</b>	
第 13 章	JSP 与 XML 文档 .....	357
13.1	JSP 的 XML 语法.....	357
13.1.1	jsp:root 元素.....	357
13.1.2	jsp:directive.page 元素.....	358
13.1.3	jsp:directive.include 元素 .....	358
13.1.4	jsp:declaration 元素 .....	358
13.1.5	jsp:scriptlet 元素 .....	359
13.1.6	jsp:expression 元素 .....	359
13.1.7	jsp:cdata 元素.....	359
13.2	JSP 与 XML 文档的映射 .....	359
13.2.1	page 编译指令 .....	359
13.2.2	taglib 编译指令 .....	361

13.2.3	include 编译指令 .....	362
13.2.4	变量声明指令 .....	362
13.2.5	Scriptlets 指令 .....	363
13.2.6	表达式 .....	365
13.3	确认 JSP 文档 .....	365
13.4	在 JSP 中使用 XML 数据源 .....	365
13.4.1	将 XML 元素转换成服务器端对象并提取数据 .....	366
13.4.2	用 XSLT 转换 XML .....	367
13.4.3	使用 JSP 生成文档 .....	367
<b>第六篇 JSP 实际开发中的应用</b>		
第 14 章	网站规划和设计 .....	371
14.1	网站的整体结构和站点的创建 .....	371
14.2	建立数据库 .....	371
14.3	公用的页面 .....	372
14.4	JavaBeans .....	375
第 15 章	用户注册登录 .....	379
15.1	用户注册系统 .....	379
15.2	用户登录系统 .....	386
第 16 章	查询商品 .....	395
第 17 章	商品分类 .....	401
第 18 章	最新、特价及缺货商品列表 .....	415
18.1	查看最新发布的商品列表 .....	415
18.2	查看特价商品列表 .....	423
18.3	查看缺货商品列表 .....	429
第 19 章	购物车 .....	431
第 20 章	用户订单 .....	439
20.1	订单信息确认 .....	439
20.2	列出所有未处理的订单列表 .....	441
20.3	查询订单 .....	444
第 21 章	论坛 .....	449
第 22 章	聊天室 .....	477
22.1	用 JSP+JavaBeans 实现的聊天室 .....	477
22.2	用 Servlet 实现的聊天室 .....	488
附录	网上资源及光盘使用说明 .....	511



# 开发专家之

# Sun ONE

## 第一篇 JSP 应用开发基础篇

在本篇中，将讲述 JSP 应用开发的基础知识。

本篇分为三章，分别讲述了 JSP 技术概论、JSP 在应用开发中的语言结构以及 JSP 运行环境的安装配置。

在第 1 章中，介绍了动态网站的基本概念技术并对 JSP 技术的工作原理进行了阐述，接下来又对 JSP 与其他开发技术进行了比较，并对 JSP 技术应用开发的未来做以展望，最后一节则是通过一个实例来介绍 JSP 技术的实际应用。

在第 2 章中，介绍了超文本标记语言，并讲述了超文本标记语言在 JSP 应用开发中的应用，接下来对 Java 语言及其在 JSP 应用开发中的应用也做了讲述。

在第 3 章中，介绍了 JSP 对运行环境的要求，接着又讲述了 JSP 开发和运行环境的安装配置。主要涉及 JDK 的安装配置、Java 集成开发环境的安装配置、数据库的安装配置。在本章第三节中则讲述了 Tomcat 在 UNIX 和 Windows 系统下的安装与配置。本章第四节中则重点讲述了安装配置中的常见问题及其解决方法。

通过对本篇的学习，将会掌握 JSP 技术在实际开发中的基本应用。





# 第 1 章 JSP 技术概论

本章主要介绍动态网站的基本概念及 JSP 技术。并对 JSP 技术的工作原理进行了阐述。接下来还要对 JSP 与其他开发技术进行比较，并对 JSP 技术应用开发的未来做一展望，最后一节则是通过一个实例来讲解 JSP 技术的实际应用。通过本章的学习，将会掌握 JSP 技术的基础知识和应用。

## 1.1 动态网站技术介绍

Internet 起源于 20 世纪 60 年代末的美国，它在近几年迅速风靡全球，其根本原因不仅在于它拥有卓越的国际通信功能，更在于它拥有巨大的信息资源。所谓的 Internet 是指由分布在全世界成千上万的计算机网络遵循一定的通讯协议，并相互联系在一起而形成的国际互联网络。也就是说，Internet 是建立和使用这些网络的人群、团体、公司以及各种网络资源的集合体。

随着网络技术的不断发展，单纯的静态页面已经不能满足发展的需要。因为静态页面是用单纯的 HTML 语言组成的，它没有交互性。因此，为了满足实际的需要，许多网页文件扩展名不再只是“.htm”、“.html”，出现了以“.php”、“.asp”、“.jsp”、“.shtml”等为后缀的网页文件，这些都是采用动态网页技术制作出来的。

在早期，动态网页主要采用 CGI 技术，CGI 即 Common Gateway Interface（公用网关接口）。您可以使用不同的程序编写适合的 CGI 程序，如 Visual Basic、Delphi 或 C/C++ 等。虽然 CGI 技术发展成熟而且功能强大，但由于编程困难、效率低下、修改复杂等缺陷，所以有逐渐被新技术取代的趋势，在这里就不再作介绍。

下面介绍几种目前颇受关注的新技术：

### ● ASP

ASP 即 Active Server Pages，它是微软开发的一种类似 HTML（超文本标记语言）、Script（脚本）与 CGI（公用网关接口）的结合体。它没有提供自己专门的编程语言，而是允许用户使用许多已有的脚本语言编写 ASP 的应用程序。

ASP 的程序比 HTML 更方便且更富有灵活性。

ASP 是在 Web 服务器端运行，运行后再将运行结果以 HTML 格式传送至客户端的浏览器。因此 ASP 与一般的脚本语言相比要安全得多。

ASP 的最大好处是可以包含 HTML 标签，也可以直接存取数据库及使用无限扩充的 ActiveX 控件，因此在程序编制上要比 HTML 方便，而且更富有灵活性。通过使用 ASP 的组件和对象技术，用户可以直接使用 ActiveX 控件，调用对象方法和属性，以简单的方式实现强大的交互功能。

但 ASP 技术也并非完美无缺，由于它基本上是局限于微软的操作系统平台，主要的

工作环境是微软的 IIS 应用程序结构，又因 ActiveX 对象具有平台特性，所以 ASP 技术要实现在跨平台 Web 服务器上工作，不是很容易。

### ● PHP

PHP 即 Hypertext Preprocessor（超文本预处理器），它是当今 Internet 上最为火热的脚本语言，其语法借鉴了 C、Java、Perl 等语言，而且只需要很少的编程知识就能使用 PHP 建立一个真正交互的 Web 站点。

它与 HTML 语言具有非常好的兼容性，使用者可以直接在脚本代码中加入 HTML 标签，或者在 HTML 标签中加入脚本代码从而更好地实现页面控制。PHP 提供了标准的数据库接口，数据库连接方便，兼容性强，扩展性强，可以进行面向对象编程。

### ● JSP

JSP 即 Java Server Pages，它是由 Sun Microsystem 公司于 1999 年 6 月推出的新技术，是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术。

JSP 和 ASP 在技术方面有许多相似之处，不过两者来源于不同的技术规范组织：ASP 一般只应用于 Windows NT/2000 平台，而 JSP 则可以在 85% 以上的服务器上运行，而且基于 JSP 技术的应用程序比基于 ASP 的应用程序易于维护和管理，所以被许多人认为是未来最有发展前途的动态网站技术。

## 1.2 JSP 技术原理

### 1.2.1 JSP 语言及其特点

在早期，开发网络数据库应用程序主要采用 CGI 技术（Common Gateway Interface 公用网关接口技术）。编写 CGI 程序可以使用不同的程序语言，如 Perl、Visual Basic、Delphi 或 C/C++ 等。虽然 CGI 技术已经发展成熟而且功能强大，但由于其编程困难、效率低下、修改复杂等缺陷，所以有逐渐被新技术取代的趋势。

在这样的背景下，新的技术纷纷面世，如 Microsoft 的 ASP（Active Server Page），Tcx 的 PHP（Hypertext Preprocessor），Sun 的 Java/Jsp/Servlet 等。其中 Sun 的 Java/Jsp/Servlet 技术被许多人认为是未来最有发展前途的动态网站技术。

JSP（JavaServer Pages）是由 Sun Microsystems 公司倡导、许多公司参与一起建立的一种动态网页技术标准。其发展的历程是这样的：

1998 年 4 月，Sun 公司发布 JSP 0.90 规范。

1999 年 1 月，Sun 公司在发布了 JSP 0.92 规范的同时，推出支持 JSP 的 Web 服务器——Java Web Server 2.0。

1999 年 11 月，Sun 公司发布 JSP 1.1 规范，同时推出 JSWDK1.0.1 和 Servlet 2.2 版。

2000 年 9 月，Sun 公司发布 JSP 1.2 规范，同时推出 Java Servlet API 2.3 版。

在开发 JSP 规范的过程中，Sun 公司与许多主要的 Web 服务器、应用服务器和开发工具供应商积极进行合作，不断完善技术。

在传统的网页 HTML 文件 (\*.htm,\*.html) 中加入 Java 程序片段(Scriptlet)和 JSP 标记(tag), 就构成了 JSP 网页 (\*.jsp)。

Web 服务器在遇到访问 JSP 网页的请求时, 首先执行其中的程序片段, 然后将执行结果以 HTML 格式返回给客户。

程序片段可以操作数据库、重新定向网页以及发送 E-mail 等, 这就是建立动态网站所需要的功能。

所有程序操作都在服务器端执行, 网络上传送给客户端的仅是得到的结果, 对客户浏览器的要求最低, 可以实现无 Plugin, 无 ActiveX, 无 Java Applet, 甚至无 Frame 的程序。

JSP 基于强大的 Java 语言, 具有良好的伸缩性, 与 Java Enterprise API 紧密地集成在一起, 在网络数据库应用开发领域具有得天独厚的优势, 基于 Java 平台构建网络程序已经被越来越多的人认为是未来最有发展前途的技术。

JSP 技术在多个方面加速了动态 Web 页面的开发。

### 1. 将内容的生成和显示进行分离

使用 JSP 技术, Web 页面开发人员可以使用 HTML 或者 XML 标识来设计和格式化最终页面。使用 JSP 标识或者小脚本来生成页面上的动态内容(内容是根据请求来变化的, 例如请求账户信息或者特定的一瓶酒的价格)。生成内容的逻辑被封装在标识和 JavaBeans 组件中, 并且捆绑在小脚本中, 所有的脚本在服务器端运行。如果核心逻辑被封装在标识和 Bean 中, 那么其他人, 如 Web 管理人员和页面设计者, 能够编辑和使用 JSP 页面, 而不影响内容的生成。

在服务器端, JSP 引擎解释 JSP 标识和小脚本, 生成所请求的内容(例如, 通过访问 JavaBeans 组件, 使用 JDBC 技术访问数据库, 或者包含文件), 并且将结果以 HTML (或者 XML) 页面的形式发送回浏览器。这有助于作者保护自己的代码, 而又保证任何基于 HTML 的 Web 浏览器的完全可用性。

### 2. 生成可重用的组件

绝大多数 JSP 页面依赖于可重用的、跨平台的组件(JavaBeans 或者 Enterprise JavaBeans TM 组件) 来执行应用程序所要求的更为复杂的处理。开发人员能够共享和交换执行普通操作的组件, 或者使得这些组件为更多的使用者或者客户团体所使用。基于组件的方法加速了总体开发过程, 并且使得各种组织在他们现有的技能和优化结果的开发努力中得到平衡。

### 3. 采用标识简化页面开发

Web 页面开发人员不一定是熟悉脚本语言的编程人员。JavaServer Page 技术封装了许多功能, 这些功能是在易用的、与 JSP 相关的 XML 标识中进行动态内容生成所需要的。标准的 JSP 标识能够访问和实例化 JavaBeans 组件, 设置或者检索组件属性, 下载 Applet, 以及执行用其他方法更难于编码或耗时的功能。

通过开发定制标识库, JSP 技术是可以扩展的。今后, 第三方开发人员和其他人员可以为常用功能创建自己的标识库。这使得 Web 页面开发人员能够使用熟悉的工具和如同标识一样地执行特定功能的构件来工作。

### 4. JSP 能提供所有 Servlets 功能

与 Servlets 相比, JSP 能提供所有的 Servlets 功能, 它比用 Println 书写和修改 HTML

更方便。此外，您可以更明确地进行分工，Web 页面设计人员编写 HTML，只需要留出空间让 Servlets 程序员插入动态部分即可。

#### 5. 健壮的存储管理和安全性

由于 JSP 页面的内置脚本语言是基于 Java 编程语言的，而且所有的 JSP 页面都被编译成为 Java Servlet，JSP 页面就具有 Java 技术的所有优点，包括健壮的存储管理和安全性。

#### 6. 一次编写，随处运行

作为 Java 平台的一部分，JSP 拥有 Java 编程语言“一次编写，随处运行”的特点。随着越来越多的供应商将 JSP 支持添加到他们的产品中，您可以使用自己所选择的服务器和工具，但并不影响当前的应用。

#### 7. JSP 的平台适应性更广

这是 JSP 比 ASP 的优越之处。几乎所有平台都支持 Java、JSP+JavaBeans，它们可以在任何平台下通行无阻。NT 下的 IIS 通过一个插件就能支持 JSP，例如 JRUN (<http://www3.allaire.com/products/jrun/>) 或者 ServletExec (<http://www.newatlanta.com/>)，著名的 Web 服务器 Apache 已经能够支持 JSP。由于 Apache 广泛应用在 NT、UNIX 和 Linux 上，因此 JSP 有更广泛的运行平台。虽然现在 NT 操作系统占了很大的市场份额，但是在服务器方面 UNIX 的优势仍然很大，而新崛起的 Linux 更是来势不小。从一个平台移植到另一个平台，JSP 和 JavaBeans 甚至不用重新编译，因为 Java 字节码都是标准的字节码与平台无关。

#### 8. Java 中连接数据库的技术是 JDBC(Java Database Connectivity)。

很多数据库系统带有 JDBC 驱动程序，Java 程序就通过 JDBC 驱动程序与数据库相连，执行查询、提取数据等操作。Sun 公司还开发了 JDBC-ODBC Bridge，用此技术 Java 程序就可以访问带有 ODBC 驱动程序的数据库，目前大多数数据库系统都带有 ODBC 驱动程序，所以 Java 程序能访问诸如 Oracle、Sybase、MS SQL Server 和 MS Access 等类型数据库。

### 1.2.2 JSP 实现原理

JSP 源文件由安装在 Web 服务器上的 JSP 引擎编译执行。

客户对 JSP 的请求直接发送给 JSP 引擎，JSP 引擎接受到请求后，按照 JSP 源代码中所规定的内容生成给客户端的响应，并把响应传递给客户端的浏览器。

让我们看一个小程序 Helloworld.jsp，运行后所得的页面效果如图 1-1 所示。

文件 Helloworld.jsp 的源代码如例程 1-1 所示。

例程 1-1

```
<%@ page language="Java" %>
<%@ page info="a hello world example" %>
<html>
<head>
<title>Hello, World</title>
</head>
<body>
```



```
<h1>
<%out.println(" Hello, World! ");%>
</h1>
</body>
</html>
```



图 1-1 helloworld.jsp 页面效果图

在 JSP 引擎上所得的 Java 文件的源代码如例程 1-2 所示。

例程 1-2

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Vector;
import org.apache.jasper.runtime.*;
import java.beans.*;
import org.apache.jasper.JasperException;

public class _0002fhelloworld_0002ejspfhelloworld_jsp_0 extends HttpJspBase {
    // begin [file="D:\\helloworld.jsp";from=(1,0);to=(1,40)]
    public String getServletInfo() {
        return "a hello world example";
    }
    // end
    static {
    }
    public _0002fhelloworld_0002ejspfhelloworld_jsp_0() {
    }
    private static boolean _jspx_initd = false;
    public final void _jspx_init() throws JasperException {
    }
    public void _jspService(HttpServletRequest request, HttpServletResponse response)
```

```

throws IOException, ServletException {
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    String _value = null;
    try {
        if (_jspx_inited == false) {
            _jspx_init();
            _jspx_inited = true;
        }
        _jspxFactory = JspFactory.getDefaultFactory();
        response.setContentType("text/html;charset=8859_1");
        pageContext = _jspxFactory.getPageContext(this, request, response,
            "", true, 8192, true);
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();
        // HTML // begin [file="D:\\helloworld.jsp";from=(0,27);to=(1,0)]
        out.write("\r\n");
        // end
        // HTML // begin [file="D:\\helloworld.jsp";from=(1,40);to=(8,0)]
        out.write("\r\n<html>\r\n<head>\r\n<title>Hello,
World</title>\r\n</head>\r\n<body>\r\n<h1>\r\n");
        // end
        // begin [file="D:\\helloworld.jsp";from=(8,2);to=(8,33)]
        out.println(" Hello, World! ");
        // end
        // HTML // begin [file="D:\\helloworld.jsp";from=(8,35);to=(12,0)]
        out.write("\r\n</h1>\r\n</body>\r\n</html>\r\n");
        // end
    } catch (Exception ex) {
        if (out.getBufferSize() != 0)
            out.clearBuffer();
        pageContext.handlePageException(ex);
    } finally {
        out.flush();
        _jspxFactory.releasePageContext(pageContext);
    }
}
}

```

JSP 源文件是由安装在 Web 服务器上的 JSP 引擎编译执行的。JSP 引擎把来自客户端



的请求传递给 JSP 源文件，然后 JSP 引擎再把对它的响应从 JSP 源文件传递给客户端。

所有的 JSP 引擎都必须支持的请求和响应协议都是 HTTP 协议，但是同一个引擎也可以支持其他的一些请求和响应协议。默认的 request 对象对应的协议是 `HttpServletRequest`，而 response 对象对应的协议则是 `HttpServletResponse`。

一个 JSP 引擎需要在传递 request 和 response 对象之前，要在 JSP 源代码中创建一个类，而 Servlet 则定义了 JSP 引擎与 JSP 源文件实现类之间的约定。例如，当使用 HTTP 协议时，`HttpServletRequest` 类就描述了 JSP 引擎与 JSP 源文件实现类之间的约定。

JSP 规范中还定义了 JSP 引擎与 JSP 页面作者之间的约定，这个约定实际上就是页面作者对 JSP 源文件中描述的动作所进行的假设，如表 1-1 所示。

表 1-1 JSP 引擎与 JSP 页面作者之间的约定

JSP 引擎调用的方法	方法说明
<code>public final void _jspInit() throws JasperException</code>	该方法可以在 JSP 页面中随意定义 JSP 页面初始化时调用该方法 当该方法调用时，在 Servlet 中的所有方法都可以使用
<code>public final void _jspDestroy() throws JasperException</code>	该方法可以在 JSP 页面中随意定义 JSP 页面结束之前时调用该方法
<code>public void _jspService (HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException</code>	该方法不能在 JSP 页面中定义 JSP 引擎自动按照 JSP 页面的内容产生这个方法 每次客户请求时，都要调用该方法

这个约定描述了一个 JSP 作者如何说明当源文件实现 `init()` 和 `destroy()` 方法时，必须执行的一些动作。在这个约定中，最重要的是 `_jspService()` 方法，这个方法是通过一个 JSP 引擎从 JSP 源文件中自动产生的。

## 1.3 JSP 与其他技术的比较

为了简明起见，下面先将 JSP、ASP、PHP 三种流行语言列表做一下比较。如表 1-2 所示。

表 1-2 JSP、ASP、PHP 性能比较

	JSP	ASP	PHP
运行速度	快	较快	较快
运行耗损	较小	较大	较大
难易程度	容易掌握	简单	简单
运行平台	绝大部分平台均可	Windows 平台	Windows/UNIX 平台

(续表)

	JSP	ASP	PHP
扩展性	好	较好	较差
安全性	好	较差	好
函数支持	多	较少	多
数据库支持	多	多	多
厂商支持	多	较少	较多
对 XML 的支持	支持	不支持	支持
对组件的支持	支持	支持	不支持
对分布式处理的支持	支持	支持	不支持
应用程度	较广	较广	较广

由上表不难看出, JSP 要比其他语言更具有优越性。

下面通过一个简单的例子来说明 JSP 与其他动态网站技术的比较。

文件 simple.jsp 的源代码如例程 1-3 所示。

例程 1-3

```
<%@ page language="Java" %>
<HTML>
<HEAD>
<TITLE> jsp </TITLE>
</HEAD>
<BODY>
<%
out.print("This is a simple jsp file!");
%>
</BODY>
</HTML>
```

运行结果如图 1-2 所示。

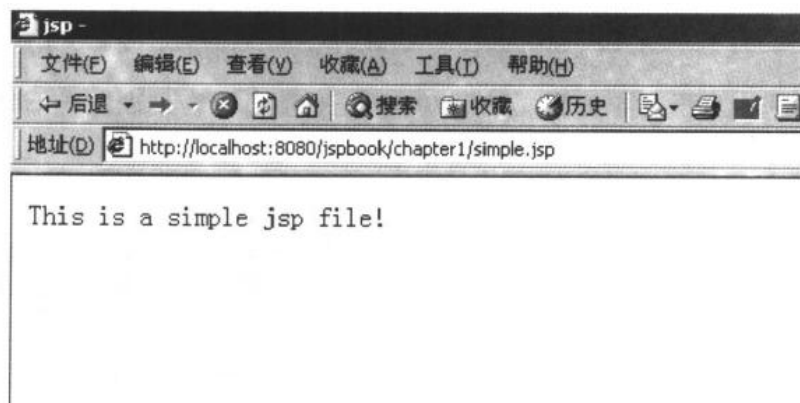


图 1-2 simple.jsp 运行结果

文件 simple.asp 的源代码如例程 1-4 所示。

例程 1-4

```
<%@ language="vbscript" %>
<HTML>
<HEAD>
<TITLE> asp </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<%
response.write "This is a simple asp!"
%>
</BODY>
</HTML>
```

运行结果如图 1-3 所示。



图 1-3 simple.asp 运行结果

文件 simple.php 的源代码如例程 1-5 所示。

例程 1-5

```
<HTML>
<HEAD>
<TITLE> php </TITLE>
</HEAD>
<BODY>
<?php
echo "This is a simple php!";
?>
</BODY>
</HTML>
```

目前最流行的动态网站技术有上述三种，即 ASP（Active Server Pages）、PHP（Hypertext Preprocessor）、JSP（Java Server Pages）。其中，PHP 和 ASP 在国内应用非常

广泛。但是，在国外，JSP 已经成为最流行的技术，特别是在 B2B 和 B2C 等电子商务领域内。

PHP 是一种跨平台的服务器端的嵌入式语言，在 UNIX 平台应用广泛。PHP 是完全免费的，也支持绝大多数数据库。但是，PHP 缺乏规模支持，而且对多层结构也缺乏支持。对于大型网站来说，应用 PHP 会造成负担过重。同时，PHP 提供的数据库接口支持不统一，这使得 PHP 不适合在电子商务领域内运行。

总的来说，JSP、ASP 两者和 PHP 相差较远，而 JSP 和 ASP 在技术方面却有许多相似之处。下面从几个角度来比较一下他们的异同点。

JSP 与 Microsoft 的 ASP 技术非常相似，两者都提供在 HTML 代码中混合某种程序代码、由语言引擎解释执行程序代码的能力。在 ASP 或 JSP 环境下，HTML 代码主要负责描述信息的显示样式，而程序代码则用来描述处理逻辑。普通的 HTML 页面只依赖于 Web 服务器，而 ASP 和 JSP 页面则需要附加语言引擎分析和执行程序代码。程序代码的执行结果被重新嵌入到 HTML 代码中，然后一起发送给浏览器。ASP 和 JSP 都是面向 Web 服务器的技术，客户端浏览器不需要任何附加的软件支持。

ASP 的编程语言是 VBScript 之类的脚本语言，JSP 使用的是 Java，这是两者最明显的区别。

此外，ASP 与 JSP 还有一个更为本质的区别：两种语言引擎用完全不同的方式处理页面中嵌入的程序代码。在 ASP 下，VBScript 代码被 ASP 引擎解释执行；在 JSP 下，代码被编译成 Servlet 并由 Java 虚拟机执行，这种编译操作仅在对 JSP 页面的第一次请求时发生。

自从微软推出 ASP (ActiveServerPage) 后，它以其强大的功能，简单易学的特点而受到广大 Web 开发人员的喜欢。但是它却有微软产品的通病，只能在 Windows 平台下使用，虽然它可以通过增加控件而在 Linux 下使用，但是其功能最强大的 DCOM 控件却不能使用。

而 Sun 公司在 Java 的基础下开发出的 JSP (Java ServerPages)，实现了动态页面与静态页面的分离，脱离了硬件平台的束缚，以及编译后运行等方式，大大提高了其执行效率而逐渐成为因特网上的主流开发工具。

### 1. 结构

JSP 和 ASP 在结构上类似，都是以“<%”和“%>”作为标记符，不同的是，在标记符之间的代码 ASP 为 JavaScript 或 VBScript 脚本，而 JSP 为 Java 代码。

JSP 将网页的表现形式和服务器的代码逻辑分开。作为服务器进程的 JSP 页面，首先被转换成 Servlet (一种服务器端运行的 Java 程序)。Servlet 支持 HTTP 协议的请求和响应。

当 JSP 被转换成纯 Java 代码，多个用户同时请求一个 JSP 页面时，应用实例化线程来响应请求。这些线程由 Web 服务器进程来管理，和 ASP 的线程管理器功能类似。同 CGI 为每个请求创建一个进程的模式比较，这种方式效率高得多。

### 2. 性能和平台无关性

与 C++、VB 等语言比较，Java 被看做是个效率不太高的语言，但它以牺牲效率换来了平台无关性，使 Java 可以在大多数操作系统上运行而不需要重新编译。

Java 的编译代码是一种字节代码，在运行时由操作系统上的一个 Java VirtualMachine (JVM) 虚拟机解释。字节代码可以在所有平台上迁移，而不需要任何改动。

Servlets 比传统的 Java 程序 (applets, Javaapp) 要快，因为它们在服务器端运行，不需要加载沉重的 GUI (HTML 的 GUI 是非常少的)。

另外，Servlets 的字节代码只有在客户请求时才执行，所以尽管当首次调用 Servlets 时会有几秒钟的加载时间，但后续的请求响应非常迅速，因为服务器已经缓存了运行的 Servlets。

当前的 JSP 服务器，都带有 Java 即时编译器 (JIT)，因此，JSP 的执行比每次都要解释执行的 ASP 代码要快，尤其是在代码中存在循环操作时，JSP 的速度要快 1 到 2 个数量级。

### 3. Session 管理

为了跟踪用户的操作状态，ASP 应用 Session 对象。JSP 使用一个叫 HttpSession 的对象实现同样的功能。

Session 的信息保存在服务器端，Session 的 ID 保存在客户机的 Cookie 中。如果客户机禁止 Cookie，Session ID 就必须放在 URL 后面。

Session 一般在服务器上设置了一个 30 分钟的过期时间，当客户停止活动后会自动失效。Session 中保存和检索的信息不能是基本数据类型 (primitive data types)，如 int、double 等，而必须是 Java 的相应的 Object (对象)，如 (Integer、Double)。

### 4. Application 管理

有时服务器需要管理面向整个应用的参数，使得每个客户都能获得同样的参数值。和 Session 一样，ASP 使用 Application 对象而 JSP 使用 ServletContext 对象，操作的方法和 Session 一样。

### 5. Server Side Includes

在服务器端的引用上，ASP 和 JSP 有着相同之处。ASP 和 JSP 都可以支持此功能的服务器 (IIS, APACHE) 上实现服务器端包含的虚拟文件。但 JSP 是将动态页面的结果包含进来，而不是包含文件代码本身。当您包含的文件在另一个服务器上时，不包含任何代码和对象是一个非常有效的功能。

### 6. Java 组件

JavaBeans 是一些完成预定义功能的封装的对象数据。JavaBeans 和 JSP 的结合与 COM 和 ASP 相比如下：

COM 对象常用来封装商业逻辑和为 ASP 页面完成高强度计算。重用的组件使得页面简单快速，因为组件由编译语言 (C++、VB) 构成，而不是解释型的 Scripting 语言 (VBScript, JavaScript)。

JavaBeans 只能用 Java 语言开发，COM 可以由符合标准对象模型的任何语言开发。另一方面，JavaBeans 更容易开发，因为一旦掌握了 Java，了解 JavaBeans 的结构就会非常容易。JavaBeans 不需要重新注册，如果开发者不需要得到对服务器的完全访问权限，这是其中一个巨大的优点。

可以创建满足商业逻辑的完整的 JavaBeans 库，让非编程人员使用库来开发动态网站。Bean 可以在服务器端管理数据库连接。



JavaBeans 符合结构化对象模型：每个 Bean 由一个不带参数的构造函数，控制它的 Servlet 可以使用内省（introspection）来设置其属性。要设置 Bean 的内置属性，必须使用带有属性名的 SetProperty 标签。如果一个属性可以设置，Bean 需要有一个 setXxxx 方法，Xxxx 用实际的属性名来代替。

#### 7. JSP 和数据库

ASP 使用 ODBC 通过 ADO 连接数据库，而 Java 通过一个叫 JDBC 的技术连接数据库。

目标数据库需要一个 JDBC 驱动程序，使得 Java 可以用标准的方式访问数据库。JDBC 不使用服务器端的数据源。只要有 JDBC 驱动程序，Java 就可以访问数据库了。

如果一个特定的数据库没有 JDBC 驱动程序，而只有 ODBC 驱动程序，Java 提供一个 JDBC-ODBC 桥来将 JDBC 调用转化为 ODBC 调用。所有的 Java 编译器都带有一个免费的 JDBC-ODBC 桥。理论上，桥可以访问任何常见的数据库产品。

结论：

JSP 模型是在 ASP 之后定义的，它借用了 ASP 的许多优点，如 Session，Application 等对象。同时 JSP 使用灵活而强大的 Java 语言，而不是低效的 Script 语言。ASP 开发者只能使用基于 Windows 平台的技术，而 Java 和 JSP 是跨平台的。

## 1.4 JSP 应用开发的未来

近年来，随着信息的全球化和国际互联网的普及化，电子商务成为互联网应用的最大热点，越来越多的企业通过使用电子商务技术来进行商业上的交易以减少成本。电子商务得到了前所未有的迅猛发展。

电子商务应用的范围也极其广泛，大致有以下几类：

#### 1. B2C

企业间的电子商务。企业通过 Internet 为消费者提供一个新兴的购物环境，即网上超市。消费者通过网络进行网上购物和网上支付等活动。

#### 2. B2B

企业与企业（Business to Business）之间，通过 Internet 或者其他网络方式进行电子商务活动。

#### 3. 企业内部电子商务

企业内部之间，通过企业内部网（intranet）的方式处理与交换商贸信息。

JSP 技术就是构建安全可靠而又高效率的电子商务应用系统的最佳选择。为了适应未来电子商务的发展需要，Sun 公司在 Java1.1 的基础上推出了 Java2.0 开发工具包，提供了 Java 企业应用编程接口（Java Enterprise API），为企业计算以及电子商务应用系统提供了有关的技术和丰富的类库。

最初，Java 是运用在浏览器和客户计算机上的，当时 Java 总被怀疑是否适合作为服务器端的开发。而现今，随着越来越多的第三方对 Java2 平台企业版（J2EE）的支持，Java 已经被广泛地用来开发企业级的服务器端应用。

JSP 技术正是基于强大的 Java 语言,因而具有良好的伸缩性,而且与 Java Eenterprise API 紧密地结合在一起,从而使得 JSP 技术成为构建电子商务系统的最佳选择。

JSP 应该是未来动态网站技术的趋势。世界上一些大型的电子商务解决方案提供商都采用了 JSP 技术。最著名的当数 IBM 公司的 E-Business,它的核心是采用了 JSP/Servlet 技术的 IBM Websphere。

## 1.5 JSP 实例讲解

在了解了 JSP 技术的工作原理以及 JSP 技术的特点优势之后,下面将会通过讲述一个简单的实例来进一步地阐述 JSP 技术。

读者可以在掌握了第二章和第三章之后,再来测试下面的实例。在这儿,只是通过这个简单的实例来向读者介绍一下 JSP 技术的实现过程和基本语法。

图 1-4 展示了一个简单的 JSP 页面,源文件对应的是 sample.jsp。

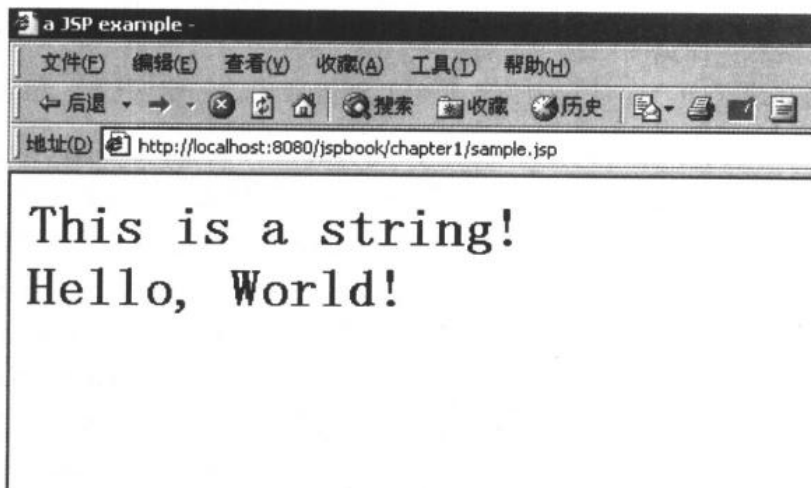


图 1-4 sample.jsp 运行效果图

文件 sample.jsp 的源代码如例程 1-6 所示。

例程 1-6

```
<%@ page language="Java" %>
<%@ page info="a JSP example" %>
<html>
<head>
<title>a JSP example</title>
</head>
<body>
<h1>
<%
String str;
str=new String("This is a string!");
```



```
%>
<%
out.println(str);
%>
<br>
<%
out.println(" Hello, World! ");
%>
</h1>
</body>
</html>
```

页面说明:

在很多 JSP 文件中会经常习惯地写上这样的说明:

```
<%@ page language="Java" %>
<%@ page info="a JSP example" %>
```

其中, `language` 属性声明了脚本语言的种类, 该页面用了 “Java”, `Info` 属性描述当前 JSP 文件的相关信息, 如页面信息等。可以在 JSP 文件中的任何地方写这种代码, 但是好的习惯是把它写在最前面。而且, 因为它是 JSP 标签, 记住一定要放在 `<html>` 前面。

**注意**

在 JSP 中对写法非常敏感, 不可以有一点错误。例如, 把 `<jsp:useBean>` 写成 `<jsp:usebean>` 那样服务器会出现错误信息。一些如类名、包名、路径名还有其他敏感的标签等千万不要写错。有一些 JSP 标签拿不准的话就去查一下 `JavaServer Pages` 语法卡片。

Tomcat 中包含的 Web 服务器的文档目录在默认状态下为 `\ tomcat\ webapps`, 主文档在默认状态下为 `index.html` 和 `index.jsp`。也就是说, 访问 `http://localhost:9090` (端口可以自己设置, 详细介绍请看下一章) 等于访问 `\ tomcat\ webapps \index.html`。用文本编辑器, 如 Windows 中的记事本 (Notepad), 创建一个文本文件 `helloworld.jsp`, 保存在 `\ tomcat\ webapps` 目录下, 其内容如上述代码所示。

在浏览器的地址栏中键入 `http://localhost: 9090/helloworld.jsp`, Tomcat 中的 Web 服务器会执行 JSP 文件中用 `<%` 以及 `%>` 括起来的 Java 程序语句, 其中 `out.print` 是将文字输出到网页, 执行结果如图 1-4 所示。

## 第2章 JSP 在应用开发中的语言结构

在本章中，将讲述超文本标记语言，并对超文本标记语言在 JSP 应用开发中的应用做以下讲述，接下来还会讲述 Java 语言，并对 Java 语言在 JSP 应用开发中的应用也予以讲述。通过对本章的学习，将会掌握 JSP 在应用开发中的语言结构。

### 2.1 超文本标记语言

#### 2.1.1 超文本标记语言语法

HTML（超文本标记语言）是网络上的通用语言，也是网络 Web 语言的基础。它是一种全置标记语言，通过嵌入代码或标记来表明文本格式。

HTML 4.0 是 HTML 最新的协议，它通过样式表、脚本、框架、嵌入对象、改进的从右到左的支持和混合方向文本、功能更强的表格、增强的表单等机制扩展了 HTML。

HTML 4.0 在设计过程中得到了国际化领域专家的帮助，因此可以用各种语言写文档，并易于在世界范围内发布。设计过程中的一个重要步骤是采用了 ISO/IEC:10646 标准作为 HTML 的字符集。它是处理国际化字符、文本方向、发音和其他语言问题最具包容性的标准。

在超文本标记语言中，经常用到的语法主要有以下内容：基本文档标记、段落标记、文字标记、格式标记、图文标记、表格、表单以及框架等。在这儿就不再详细介绍，感兴趣的读者可以参考有关介绍 HTML 的书籍资料。

层叠样式表 CSS 是对 HTML 的补充。目前，CSS 有两个官方标准：CSS1 和 CSS2。因为 CSS2 标准要比 CSS1 更加成熟，所以本节只介绍 CSS2 推荐标准实现的样式表。

将 CSS 加入网页的方式有三种：

- 内联样式表（Inline Style Sheet）

最初将样式表加入到网页中的方法就是在 Style 中定义文字的属性，如：`<Element Style="Property:Value; Property:Value;...;Property:Value;"></Element>`。像这样的定义 Style 属性的方法称为内联样式表（Inline Style）。内联样式表可以应用到 HTML 当中从<A>到<VAR>的任何元素，而且只对您定义的文字作修改。

- 外部样式表（Linked Style Sheet）

将样式表加入到 HTML 中的最简单的方法就是使用外部样式表，也可称为链接样式表。外部样式表是包含了样式格式信息的一个单独的文件。

下面是一个样式表的例子：

```
H1 {
```

```
color : Blue;
font-size : 10px;
font-style : italic;
font-weight : bold;
text-decoration : overline;
}
```

把上述内容存为 style1.css 文件。

使用链接样式表的方法是在超文本的文件头中加入一个链接,该链接指向一个样式表文件。必备的属性有 Href="Url" 和 Rel="StyleSheet", Title 和 Type 是可选的。如果使用 Title 和 Type, Title 必须与样式表中 Title: property 中的 Title 一致, Type 必须设定为 "text/css"。

#### ● 嵌入样式表 (Embedded Style Sheet)

使用嵌入式样式表时,要先在文件头<HEAD>中定义一个<Style> </Style>样式块。这个块包含了一系列定义,每个定义都设定了一个或一组 HTML 标签属性。

Style 的必备属性是 Type,并且值为 text/css。有些浏览器不支持 CSS,因此会将 Style 的标签显示出来。防止这种情况出现的方法是在 HTML 中将样式表用标注方式括起来,这样不支持 CSS 的浏览器将不理睬这段内容,而支持 CSS 的浏览器会使用这段内容。具体使用方法是在 Style 开始标签之后,样式表内容之前加上 "<!--",在 Style 结束标签之前,样式表内容之后加上 "-->"。

一个样式表规则包括两部分:一个用来识别一个或一组 HTML 标签的选择器 selector;一个用来设定样式属性的声明 declaration。

定义语法如下: selector {property:value; property:value; ... }

每一个样式规则必须以一个或一组选择器作为开始,后面跟着花括号 ({}),然后是一组声明。每一个声明以分号做结束,以冒号分隔为属性和值两部分,最后是以 () 结尾。

使用嵌入式样式表会使文件中所有<H1>标签中文字显示为蓝色,除了包含内联式样式表的<H1>中的文字。这表示内联样式表在与嵌入样式表同时使用时,前者会具有更高的优先级。

#### 1. 类 CLASS

所有的 HTML 标签都可以使用样式表来定义显示特性。有时您在编辑网页时每隔一段都要用到相同的特定字体,这些字体如果都用样式表来定义就显得太麻烦了。我们可以将样式表格式分为几类,在用到时只要调用一下就行了。比如,您想让主体文字的第 1 段用绿色显示,第 2 段用紫色显示,而第 3 段还用绿色显示,您能做到吗?这种情况下“类”将发挥作用。您可以将段落 P 分成多种不同的类别,每一类应用不同的样式表说明,也可以理解为将一种标签分成几类,“类”实际上是 HTML 当中用来描述标签元素的一个属性。



类名以 (.) 开头。一个类就是一组标签元素,这些标签存在于整个超文本文件当中,所有这些标签都具有与类中声明标签相同的属性。

#### 2. 标识 ID

和类 CLASS 一样,标识 ID 也是 HTML 当中可以定义标签的一个属性。通常类是用

来定义一组有共同功能或格式的标签，而标识是用来定义某一个特定的标签。也就是说许多标签都使用同一个类，而任何一个文件中只能有一个标签能使用某一标识。标识名前面要以符号（#）开始。

通过使用 ID，您可以将多个样式表的属性比如字体颜色、字体修饰等应用到一个标签的属性中去，即让该标签中的文字样式与 ID 中定义的一样。



CLASS 和 ID 名必须以字母开头，而且名称中只能含有字母、数字和横线'-'。

### 3. DIV 和 SPAN

CLASS 和 ID 除了定义样式以外，似乎没有太多用处。而<DIV>和<SPAN>则是做网页老手常用的标签，它们可以用来调用所有的样式表属性。<DIV>定义的是包含文本和 HTML 标签的一个块，块之间用换行标签间隔，<DIV>对文字显示整行都会起作用，只有通过换行才能改变属性。<SPAN>是一个内联标签元素，可以存在于内容的任何地方，它的作用范围只是<SPAN>内的文字，而不是整个一行。

## 2.2.2 动态超文本标记语言

动态 HTML (DHTML) 针对 Netscape 和 Microsoft 有着不同的定义。因为 DHTML 实际上涵盖了几种技术，每一种都有不同的标准。主要包括的技术有：层叠样式表单 (CSS)；层叠样式表单定位 (CSS-P)；文档对象模型 (DOM) 以及客户端的 Script 编辑。

层叠样式表单 (CSS) 将在后面的章节中讲述，在这儿不做介绍。

层叠样式表单定位 (CSS-P)，这是另一个标准，用来指定 html 元素在页面上的位置。它的目的是扩展 CSS1，使元素可以在页面上精确定位。

文档对象模型 (DOM) 也是 DHTML 中最重要的。由于 Navigator 和 Internet Explorer 的文档对象模型有所不同，所以一个统一的 Script 对象的 DOM 标准很难形成。每个平台安装已经存在的文档对象模型越多，达成一致意见就会越难，它也许需要把一些属性压到标准水平，才能达成 Script 的共同点。

客户端的 Script 编辑是指将来的 HTML 标准允许的使用任何浏览器支持的语言。Navigator 只支持 JavaScript，而 Internet Explorer 支持自己的 JavaScript (JScript) 和 VBScript，因为要跨平台支持，所以 JavaScript 就成为客户端 Script 编写语言的最好选择。

JavaScript 语言和 Java 语言是相关的，虽然 JavaScript 与 Java 有紧密的联系，但却是两个公司开发的不同的两种产品。Java 是 Sun 公司推出的新一代面向对象的程序设计语言，特别适合于 Internet 应用程序开发；而 JavaScript 是 Netscape 公司的产品，其目的是为了扩展 Netscape Navigator 功能，而开发的一种可以嵌入 Web 页面中的基于对象和事件驱动的解释性语言，它的前身是 Live Script，而 Java 的前身是 Oak 语言。

下面对两种语言的异同做如下比较。

两者的区别体现在：

#### 1. 它们是两个公司开发的不同的两种产品

Java 是 Sun 公司推出的新一代面向对象的程序设计语言，特别适合于 Internet 应用程

序开发；而 JavaScript 是 Netscape 公司的产品，其目的是为了扩展 Netscape Navigator 功能，而开发的一种可以嵌入 Web 页面中的基于对象和事件驱动的解释性语言。

## 2. 基于对象和面向对象

JavaScript 是基于对象的，而 Java 是面向对象的，即 Java 是一种真正的面向对象的语言，即使是开发简单的程序，也必须设计对象。JavaScript 是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象和事件驱动的编程语言。因而它本身提供了非常丰富的内部对象供设计人员使用。

## 3. 解释和编译

两种语言在其浏览器中所执行的方式不一样。Java 的源代码在传递到客户端执行之前，必须经过编译，因而客户端上必须具有相应平台上的仿真器或解释器，它可以通过编译器或解释器实现独立于某个特定的平台编译代码的束缚。JavaScript 是一种解释性编程语言，其源代码在发往客户端执行之前不需经过编译，而是将文本格式的字符代码发送给客户，由浏览器解释执行。

## 4. 变量

两种语言所采取的变量是不一样的。

Java 采用强类型变量检查，即所有变量在编译之前必须作声明。

例如：

```
String name;  
int age;  
boolean isMarried;  
name="blueriver";  
age=22;  
isMarried=false;
```

其中：

name="blueriver"说明是一个字符串；

age=22 说明是一个整数值；

isMarried=false 说明是一个布尔值。

而在 JavaScript 中，变量在使用前不需要声明，而是解释器在运行时检查其数据类型。

例如：

```
name="blueriver";  
age=22;  
isMarried=false;
```

变量 name、age、isMarried 在使用前不需要声明，在运行时，name="blueriver"说明是一个字符串，age=22 说明是一个整数值，isMarried=false 说明是一个布尔值。

## 5. 代码格式不一样

Java 是一种与 HTML 无关的格式，必须通过像 HTML 中引用外媒体那样进行装载，其代码以字节代码的形式保存在独立的文档中。

JavaScript 的代码是一种文本字符格式，可以直接嵌入 HTML 文档中，并且可动态装载。编写 HTML 文档就像编辑文本文件一样方便。



### 6. 嵌入方式不一样

在 HTML 文档中，两种编程语言的标识不同。

- JavaScript 使用 `<script>...</script>` 来标识
- Java 使用 `<applet> ... </applet>` 来标识

### 7. 静态绑定和动态绑定

Java 采用静态联编，即 Java 的对象引用必须在编译时进行，以便使编译器能够实现强类型检查。

JavaScript 采用动态联编，即 JavaScript 的对象引用在运行时进行检查，如不经编译则无法实现对象引用的检查。

## 2.2 JSP 开发中的 Java 应用

本节首先介绍了 JSP 开发中的 Java 语法，然后列举了 Java 语法的运用实例。

### 2.2.1 在 JSP 开发中的 Java 语法

下面详细地为您介绍 Java 语法的结构及其运用。

#### 2.2.1.1 程序结构

Java 语言的源程序代码由一个或多个编译单元（compilation unit）组成，每个编译单元只能包含以下内容（空格和注释除外）：

- 一个程序包语句（package statement）
- 入口语句（import statements）
- 类的声明（class declarations）
- 界面声明（interface declarations）

每个 Java 的编译单元可以包含多个类或界面，但是每个编译单元最多只能有一个类或者界面是公共的。

#### 2.2.1.2 注释

Java 中可以采用三种注释方式：

- `//` 用于单行注释 注释从 `//` 开始，终止于行尾。
- `/*` 用于多行注释...`*/` 注释从 `/*` 开始，到 `*/` 结束，且这种注释不能互相嵌套。
- `/**` Java 所特有的文档注释...`*/` 它以 `/**` 开始，到 `*/` 结束。

这种注释主要是为支持 JDK 工具 javadoc 而采用的。javadoc 能识别注释中用标记 @ 标识的一些特殊变量，并把 doc 注释加入它所生成的 HTML 文件。这种注释一般放在一个变量或函数定义前，指示在任何自动生成文档系统中调入。这个注释都是声明条目的描述。

#### 2.2.1.3 标识符

变量、函数、类和对象的名称都是标识符，程序员需要标识和使用的东西都需要标识

符。在 Java 语言里,标识符以字符或\_、\$开头,后面可以包含数字,标识符的大小写有区别,但没有长度限制。

Java 语言有 51 个关键词和保留字,如表 2-1 所示。

表 2-1 Java 语言的关键词和保留字

abstract	const	final	instanceof	private	switch
void	boolean	continue	finally	int	protected
synchronized	volatile	break	default	float	interface
public	this	while	byte	do	for
long	return	throw	case	double	goto
native	short	throws	catch	else	if
new	static	transient	char	extends	implements
null	strictfp	true	class	import	package
super	try				

其中, goto 与 const 是保留字。尽管在 Java 语言中这两个保留字不表示任何特定意思,但是程序员最好不要使用这两个保留字作为标识符。

例如:

```
football      //合理
implement     //合理, 记住关键词是 implements, 而不是 implement。
3_person_is_cleve //不合理, 标识符以字符或_、$开头。
strictfp      //不合理, strictfp 是关键词。
```

#### 2.2.1.4 数据类型

根据 Sun 公司的白皮书, Java 有以下几种基本的数据类型, 分别是:

- boolean            • int
- char                • long
- byte                • float
- short               • double

这八种数据类型所占有的字节大小如表 2-2 所示:

表 2-2 数据类型占用的字节大小列表

类 型	字 节 数	类 型	字 节 数
boolean	1	char	16
byte	8	short	16
int	32	long	64
float	32	double	64

其中, boolean 型数据只有 true、false 值。

四种整数型数据类型 byte、short、int、long 的范围如表 2-3 所示。



表 2-3 数据类型的范围列表

类 型	字 节 数	类 型	字 节 数
byte	8	-2E7	2E7-1
short	16	-2E15	2E15-1
int	32	-2E31	2E31-1
long	64	-2E63	2E63-1

boolean 型数据只有 true、false 值。

例如：

```
boolean    regAttempt = false;
boolean    login = true;
```

char 型数据是无符号的数字型数据类型。

例如：

```
char ii='\u4567';
```

float 型数据，例如：

```
1.23f, float.NaN, float.NEGATIVE_INFINITY
```

double 型数据，例如：

```
1.23D, double.NaN, double.NEGATIVE_INFINITY
```

String 型数据，例如：

```
String logname,realname,passwd1,passwd2,email,gender,phone;
String errorMessage = "";
```

下面再介绍一种数据类型——数组。

数组是有序数据的集合，数组中的每个元素具有相同的数、数组名和下标，并用来惟一地确定数组中的元素。

### 1. 一维数组

#### ● 一维数组的定义

一维数组的定义方式为：

```
type arrayName[];
```

其中类型 (type) 可以为 Java 中任意的数据类型，arrayName 为一个合法的标识符，[] 指明该变量是一个数组类型变量。

例如：

```
int intArray[];
String[] hobbies;
```

#### ● 一维数组的初始化

在定义数组的同时进行初始化：

例如：

```
int a[]={1, 2, 3, 4, 5};
```

### 2. 多维数组

#### ● 二维数组的定义

二维数组的定义方式为：

```
type arrayName[][];
```

例如:

```
int intArray[][];
```

- 二维数组的初始化有两种方式:

- (1) 直接对每个元素进行赋值。
- (2) 在定义数组的同时进行初始化。

如: `int a[][]={{2, 3}, {1, 5}, {3, 4}};`

定义了一个  $3 \times 2$  的数组, 并对每个元素赋值。

### 2.2.1.5 常量与变量

#### 1. 常量

Java 中的常量值是用文字串表示的, 它区分为不同的类型。

例如:

整型常量: 123

实型常量: 1.23

字符常量: 'a'

布尔常量: true

字符串常量: "I love China"

与 C、C++ 不同, Java 中不能通过 `#define` 命令把一个标识符定义为常量, 而是用关键字 `final` 来实现。

例如:

```
final double PI=3.14159。
```

#### 2. 变量

变量是 Java 程序中的基本存储单元, 它的定义包括变量名、变量类型和作用域几个部分。

- 变量名是一个合法的标识符, 它是字母、数字、下划线或美元符 "\$" 的序列, Java 对变量名区分大小写, 变量名不能以数字开头, 而且不能为保留字。

例如:

以下变量为合法的变量名:

logname、w\_1、dollar\$

以下变量为非法的变量名:

2book、wangxining#、class (保留字)。

值得注意的是: 变量名应具有一定的含义, 以增加程序的可读性。

- 变量类型可以为上面所讲述的任意一种数据类型。
- 变量的作用域指明可访问该变量的一段代码。

声明一个变量的同时也就指明了变量的作用域。按作用域来分, 变量又有下面几种: 局部变量、类变量、方法参数、例外处理参数。

局部变量在方法或方法的一块代码中声明, 它的作用域为它所在的代码块(整个方法或方法中的某块代码)。

类变量在类中声明, 而不是在类的某个方法中声明, 它的作用域是整个类。

方法参数传递给方法, 它的作用域就是这个方法。

例外处理参数传递给例外处理代码, 它的作用域就是例外处理部分。

## 2.2.1.6 流程控制语句

下面分别介绍 Java 中的流程控制语句。

## 1. 分支语句

分支语句提供了一种控制机制,使得程序的执行可以跳过一些语句不执行,而转去执行特定的语句。

## ● 条件语句 if-else

if-else 语句根据判定条件的真假来执行两种操作中的一种,它的格式为:

```
if (boolean-expression)
{
    statement1;
}
else
{
    statement2;
}
```



使用 if-else 需要注意以下几点:

布尔表达式 `boolean-expression` 是任意一个返回布尔型数据的表达式(这比 C、C++ 的限制要严格)。

每个单一的语句后都必须有分号。

语句 `statement1`, `statement2` 可以为复合语句,这时要用大括号 `{}` 括起。建议对单一的语句也用大括号括起,这样程序的可读性强,而且有利于程序的扩充(可以在其中填加新的语句)。`{}` 外面不加分号。

`else` 子句是任选的。

若布尔表达式的值为 `true`,则程序执行 `statement1`,否则执行 `statement2`。

## ● 多分支语句 switch

switch 语句根据表达式的值来执行多个操作中的一个,它的一般格式如下:

```
switch (expression)
{
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
    .....
    case valueN:
        statementN;
        break;
    default:
        defaultStatement;
```

```
break;  
}
```

例如:

```
<%!  
public static String category(int id)  
{  
    try  
    {  
        switch(id)  
        {  
            case 1:  
                return "计算机类";  
                //break;  
            case 2:  
                return "英语类";  
                //break;  
            default:  
                return "其他类";  
                //break;  
        }  
    }  
    catch(Exception e)  
    {  
    }  
    return "null";  
}  
%>
```

## 2. 循环语句

Java 中提供的循环语句有: while 语句, do-while 语句和 for 语句, 下面分别来做介绍。

### ● while 语句

一般格式为:

```
while (boolean-expression)  
{  
    statement;  
}
```

例如:

```
int i=0;  
while (i<10)  
{  
    i ++;  
    System.out.println(i);  
}
```

//输出结果为 1、2、3、.....、8、9。

### ● do-while 语句

一般格式为:

```
do
{
    statement;
}
while (boolean-expression);
```



do-while 语句首先执行循环体, 然后计算终止条件, 若结果为 true, 则循环执行大括号中的语句, 直到布尔表达式的结果为 false。

与 while 语句不同的是, do-while 语句的循环体至少执行一次, 这是 do-while 语句的特点。

例如:

```
int i=0;
do
{
    i++;
    System.out.println(i);
}
while (i<10);
//输出结果为 1、2、3、.....、8、9、10。
```

#### ● for 语句

一般格式为:

```
for (initialization; boolean-expression; iteration)
{
    statement;
}
```



for 语句执行时, 首先执行初始化操作, 然后判断终止条件是否满足, 如果满足, 则执行循环体中的语句, 最后执行迭代部分。完成一次循环后, 重新判断终止条件。

可以在 for 语句的初始化部分声明一个变量, 它的作用域为整个 for 语句。

for 语句通常用来执行循环次数确定的情况(如对数组元素执行操作), 也可以根据循环结束条件执行循环次数不确定的情况。

在初始化部分和迭代部分可以使用逗号语句, 来进行多个工作。逗号语句是用逗号分隔的语句序列。例如:

```
for(i=0, j=10; i<j; i++, j--){
    .....
}
```

初始化、终止以及迭代部分都可以为空语句, 三者均为空的时候, 相当于一个无限循环。



例如:

```
for(int i=0; i<10; i++)
{
    System.out.println(i);
}
//输出结果为 1、2、3、.....、8、9。
```

## 2.2.2 JSP 开发中的 Java 应用

上一节学习了 Java 的语法,这一节我们来学习它在 JSP 开发中的应用。

### 2.2.2.1 Java 源程序的编译

每一个 JSP 文件经过编译运行后,其实就是一个 Servlet 文件,而 Servlet 文件其实又是一个 Java 语言文件。

例如,在 JSP 编译后所对应的 Servlet 中,语言结构就是一个 Java 语言所得的类。下面的代码就是一个 Servlet 文件,实际上也是一个 Java 源文件。

```
//程序包语句
package chapter1;
//入口语句
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
//类的声明
public class Servlet2 extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化全局变量*/
    public void init() throws ServletException {
    }
    /**响应 HTTP Get 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Servlet2</title></head>");
        out.println("<body>");
        out.println("<p>The servlet has received a GET. This is the reply.</p>");
        out.println("</body></html>");
    }
    /**释放资源*/
    public void destroy() {
    }
}
```

上述的 Java 源程序代码被编译后,便产生了 Java 字节代码。Java 的字节代码由一个

不依赖于机器的指令组成，这个指令能被 Java 的运行系统有效地解释。Java 的运行系统工作起来如同一台虚拟机。在当前的 Java 实现中，每个编译单元就是一个以 .java 为后缀的文件。每个编译单元有若干个类，编译后，每个类生成一个 .class 文件。 .class 文件是 Java 虚拟机能够识别的代码。

#### 2.2.2.2 Java 中的异常处理

在 Java 语言中，异常处理是非常重要的部分。同样在 JSP 应用开发中，异常处理也是非常重要的。

Java 对“异常”的处理是非常复杂的，关于这一方面的书籍也很多，在这儿只是简要介绍以下基本概念，有兴趣的读者可以参阅其他相关书籍。

##### 1. “异常”的基本概念

“异常”简单讲就是非正常，指的是程序运行时，出现的打断程序正常执行的事件。在用传统的语言编程时，程序员只能通过函数的返回值来发出错误信息。这易于导致很多错误，因为在很多情况下需要知道错误产生的内部细节。通常，用全局变量 `errno` 来存储“异常”的类型，但这容易导致误用，因为一个 `errno` 的值有可能在被处理前被另外的错误覆盖掉。即使最优秀的 C 语言程序，为了处理“异常”情况，也常求助于 `goto` 语句。

Java 对“异常”的处理是面向对象的。一个 Java 的 `Exception` 是一个描述“异常”情况的对象。当出现“异常”情况时，一个 `Exception` 对象就产生了，并放到产生这个“异常”的成员函数里。

Java 的“异常”处理是通过 5 个关键词来实现的：`try`，`catch`，`throw`，`throws` 和 `finally`。用 `try` 来执行一段程序，如果系统抛出（`throws`）一个“异常”，您可以通过它的类型来捕捉（`catch`）它，或最后（`finally`）由默认处理器来处理。下面是“异常”处理程序的基本形式。

```
try{
    //程序块
}
catch(ExceptionType1e)
{
    //对 ExceptionType1 的处理
}
catch(ExceptionType2e)
{
    //对 ExceptionType2 的处理 throw(e);
    //再抛出这个“异常”
}
finally
{
}
```

在“异常”类层次的最上层有一个单独的类叫做 `Throwable`。这个类用来表示所有的“异常”情况。每个“异常”类型都是 `Throwable` 的子类。

`Throwable` 有两个直接的子类：

一类是 `Exception`，是用户程序能够捕捉到的“异常”情况。我们将通过产生它的子

类来创建自己的“异常”。

另一类是 `Error`，它定义了那通常无法捕捉到的“异常”。要谨慎使用 `Error` 子类，因为它们通常会导致灾难性的失败。

## 2. try 与 catch

Java 中异常处理机构包括两方面的内容：发出异常信号和建立异常处理程序。发出异常信号最简单的办法是使用 `throw` 语句和一个异常对象实例。

例如：

```
Throw new some Exception( );
```

如想获得异常处理，可使用 `try` 语句和 `catch` 语句。

语法规则：

```
try
{
    //接受监视的程序块
    //一旦该程序块中的程序出现异常，系统将
    //程序交给后面适当的 catch 程序块处理
}
catch (要处理的异常 exception)
{
    //如果发生的是这种异常的话
    //系统将程序交该区块的程序处理
}
```

## 3. 多个 catch 子句

在某些情况下，同一段程序可能产生不止一种“异常”情况。您可以放置多个 `catch` 子句，其中每一种“异常”类型都将被检查，第一个与之匹配的就会被执行。如果一个类和其子类都有的话，应把子类放在前面，否则将永远不会到达子类。

例如：

```
try
{
    //接受监视的程序块
    //一旦该程序块中的程序出现异常，系统将
    //程序交给后面适当的 catch 程序块处理
}
catch (要处理的第一种异常 exception1)
{
    //如果发生的是这种异常的话
    //系统将程序交该区块的程序处理
}
catch{ 要处理的第二种异常 exception2}
{
    //如果发生的是这种异常的话
    //系统将程序交该区块的程序处理
}
```

#### 4. finally 语句

程序中有些程序段，不管发生什么异常情况都必须执行，如关闭已打开的文件等。这就需要将其放入 finally 语句的程序块中。

finally 语句通常与 catch 语句在一起使用，finally 语句可按下面的方式使用。

```
try
{
    //程序段
}
finally
{
    //不论意外是否发生，都会执行该段程序
}
```

例如：

```
public class Application1 {
    public static void main(String args[])
    {
        try
        {
            int a,b,c;
            a=1;
            b=0;
            c=a/b;
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
        finally
        {
            System.out.println("This is finally statement.");
        }
    }
}
```

输出结果：

```
/ by zero
This is finally statement.
```

#### 5. throw 语句

throw 语句用来标明一个成员函数可能抛出的“异常”。

首先得到一个 Throwable 的实例的控制柄，通过参数传到 catch 子句，或者用 new 操作符来创建一个。

程序会在 throw 语句后立即终止，它后面的语句执行不到，然后在包含它的所有 try 块中从里向外寻找含有与其匹配的 catch 子句的 try 块。

例如：

```
public class Application1 {
```

```
public static void main(String args[])
{
    try
    {
        try
        {
            throw new NullPointerException("");
        }
        catch(NullPointerException ne)
        {
            System.out.println(ne.getMessage());
            throw ne;
        }
    }
    catch(NullPointerException e)
    {
        System.out.println(e.getMessage());
    }
    finally
    {
        System.out.println("This is finally statement.");
    }
}
```

## 6. throws 语句

throws 用来标明一个成员函数可能抛出的各种“异常”。

对大多数 Exception 子类来说, Java 编译器会强迫您声明在一个成员函数中抛出的“异常”的类型。

如果“异常”的类型是 Error 或 RuntimeException, 或它们的子类, 这个规则不起作用。如果您想明确地抛出一个 RuntimeException, 您必须用 throws 语句来声明它的类型。

例如:

```
public class Application1 {
    static void proc() throws NullPointerException
    {
        throw new NullPointerException("NullPointerException");
    }
    public static void main(String args[])
    {
        try
        {
            proc();
        }
        catch(NullPointerException e)
        {
            System.out.println(e.getMessage());
        }
    }
}
```



```
    }  
    finally  
    {  
        System.out.println("This is finally statement.");  
    }  
}  
}
```

输出结果:

```
NullPointerException  
This is finally statement.
```



# 第 3 章 JSP 运行环境的安装配置

在本章中，将讲述 JSP 对运行环境的要求，接着要讲述 JSP 开发和运行环境的安装配置，主要涉及了 JDK 的安装配置、Java 集成开发环境的安装配置、数据库的安装配置三个方面。

在本章的第三节中将会讲述 Tomcat 在 UNIX 和 Windows 系统下的安装与配置。在本章的第四节中则是重点地讲述了安装配置中常见问题及其解决方法。通过对本章的学习，将会掌握 JSP 运行环境的安装配置。

## 3.1 JSP 对运行环境的要求

下面首先将讲述一下 JSP 对运行环境的一些要求。本节中将从硬件条件、操作系统、软件环境三种基本的方面加以阐述。

### 3.1.1 对硬件条件的要求

表 3-1 为对硬件要求的最低配置，建议使用当前中等以上计算机配置为最佳。

表 3-1 对硬件条件的要求

CPU	使用 586 或更高级别的处理器
内存	64MB 及其以上
硬盘	100MB 以上
光驱	8 倍速以上 CD-ROM
软驱	
显示器	
其他计算机设备	

### 3.1.2 对操作系统的要求

JSP 能够运行在目前绝大多数的操作系统上，目前在普通用户中用到的系统绝大多数为 Windows 系列和 Linux 系列。表 3-2 列出了当前常用的一些操作系统。

表 3-2 对操作系统的要求

操作系统	Microsoft Windows98, Windows NT 4.0, Windows 2000 professional, windows 2000 server 等; UNIX 系统; Red Linux; Turbo Linux;
------	---

### 3.1.3 对软件环境的要求

我们知道, 如果一个网站正常运行, 除了必要的硬件外, 还要有相应的服务器软件, 如 Microsoft 的 IIS (Internet Information Server) 等。因此我们需要安装如表 3-3 所示的软件, 以便使得网站正常运行。

表 3-3 对软件环境的要求

Web 浏览器	任何支持 HTML TABLE 和 FORM 元素的浏览器, 浏览器应支持如下 HTML: Frames Tables 客户端的图片映射 文件上载 (可选) Javascript Cookie (可选)
Web 服务器	BEA Weblogic 6.0 IBM WebSphere Server Apache Tomcat Apache Web Server Jserv Allaire Jrun 3.0 Apache Tomcat Apache Web Server Jserv ATG Dynamo Application Bluestone Sapphire Caucho Resin EasyThings Web Server ExOffice Intalio Gefion Software WAICollRunner Gefion Software Lite GemStone J4.0 Inprise Application Server 4 IPlanet Application Server Server 6.0

(续表)

	IPlanet Web Server 4.1 Java Web Server 2.0 JSWDK1.0.1 Lutris Technologies Enhydra Server 3.0 Mort Bay Jetty Novocode NetForge Oracle 8i Jserver Orion Application Server Paralogic WebCore Pramati Server Secant Technologies Extreme Server 3.5 Servartec iServer 1.8 Silverstream Application Server 3.0 Unify eWave ServletExec VqServer W3C Jigsaw2.2.1 Zeus Web Server
数据库	Oracle Sybase Microsoft SQL server Informix DB2 MySQL Microsoft Access
Java Developer Kit (JDK)	jdk1_2_2-001-win.exe

## 3.2 JSP 开发和运行环境的安装配置

下面将讲述 JSP 开发和运行环境的安装与配置。本节中将从 JDK 的安装配置、应用程序服务器的安装配置、Java 集成开发环境的安装配置三个方面加以阐述。

### 3.2.1 JDK 的安装配置

#### 1. JDK 的安装

JDK 的安装过程非常简单。在这里，只对 JDK 的安装做一简单的说明。下面以安装 jdk1.3 为例进行说明。

双击 j2sdk1\_3\_0rc2-win.exe 文件进行安装，首先出现版权协议说明界面，单击【Yes】



按钮，接受版权协议后，选择要安装的路径，如 d:\jdk1.3，然后单击【OK】按钮，再单击【NEXT】按钮继续安装。选择所有的部件，然后单击【NEXT】按钮，进行安装。安装完毕后，单击【FINISH】按钮结束安装。

## 2. JDK 的配置

除了配置标准的 Java 环境变量外，还应该必须设置 PATH 环境变量与 CLASSPATH 环境变量。

### ● PATH 命令行路径参数的设置

所创建的脚本示例通常都会在创建或运行某个示例时，自动地设置 PATH 环境变量。PATH 是用来指定可执行应用程序的路径。

设置的方法如下：

```
PATH=%PATH%;D:\jdk1.3
```

在上述设置中，JDK 被安装在 D 盘下。若是安装在其他的盘下，上述代码只要做相应的改变即可。

### ● CLASSPATH 类路径参数的配置

所创建的脚本示例通常都会在创建或运行某个示例时，自动地设置 CLASSPATH 环境变量。CLASSPATH 是用来指定在初始化 JRE（Java Run Environment Java 运行环境）时需要使用的类文件与文档文件（也就是 JAR 文件）。

设置的方法如下：

```
set CLASSPATH=D:\jdk1.3\lib\Tools.jar;D:\jdk1.3\lib\dt.jar
```

在上述设置中，JDK 被安装在 D 盘下。若是安装在其他的盘下，上述代码只要做相应的改变即可。

## 3.2.2 应用程序服务器的安装配置

目前在企业中，使用的服务器大多数是 BEA 公司的 WebLogic 和 IBM 公司的 WebSphere。而 Tomcat 一般适合小型企业或个人使用。

下面，以 WebSphere 为例子讲述一下应用程序服务器的安装过程。

WebSphere Application Server 提供从电子商务网站的构建、发布到管理能力。标准版提供了开放、标准的平台和工具，以加速向电子商务的迁移。新加入的 DB2 Universal Database5.2 支持，使开发者能够设计更强大有效的应用。WebSphere Application Server 包括 Servlet 运行引擎（RuntimeEngine），高性能的数据库连接程序（提供预联接，会话和状态管理的应用服务，以及支持 XML（eXtensibleMark-upLanguage）文档结构）。

### 1. 硬件要求

CPU: Pentium II 266MHz 以上

内存: 64MHz 以上

硬盘: 2GB 以上

### 2. 软件要求

操作系统: Windows NT Server 4.0 中文版 及 Service Pack 3

浏览器: 要求支持 Java 1.1 及以上版本的浏览器

## 3. 安装过程

## 步骤

(1) 在 CD 驱动器中插入 CD-ROM。如果在系统中 autorun 不能用, 在 CD 驱动器的根目录下运行 IBMWebAS\_NT\_202\_SE.exe。出现如图 3-1 所示的界面。

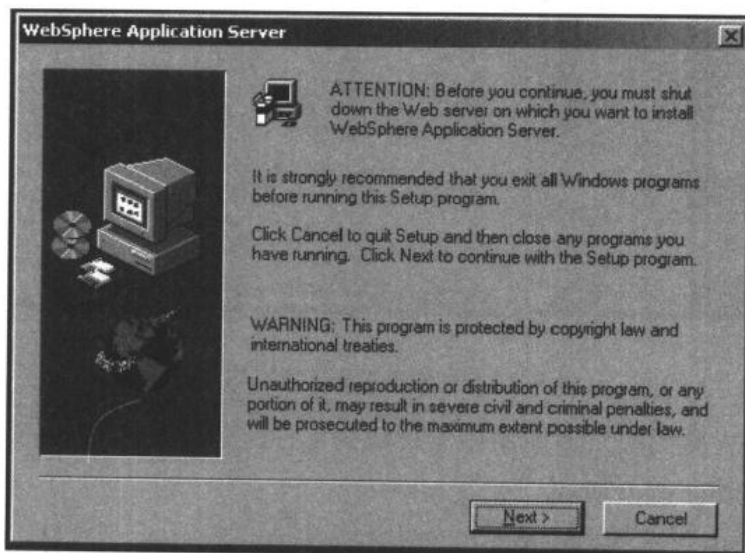


图 3-1 WebSphere Application Server 安装

(2) 单击【NEXT】按钮后, 出现如图 3-2 所示的界面。

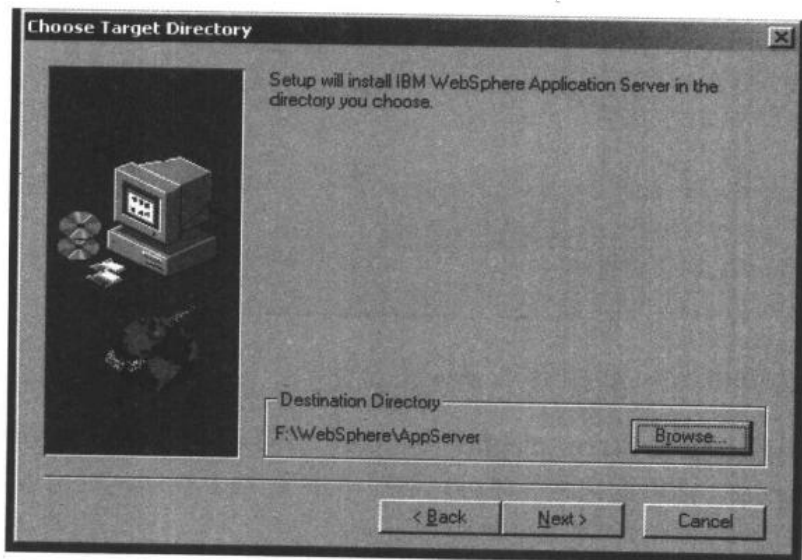


图 3-2 选择安装路径

(3) 选择安装路径, 单击【NEXT】按钮后, 出现如图 3-3 所示的界面。

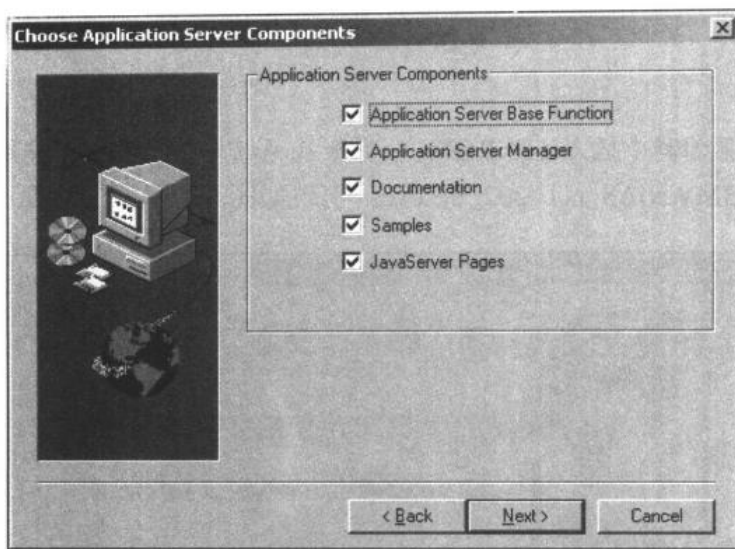


图 3-3 应用程序服务器所需的部件

(4) 选中要安装的应用程序服务器所需的部件，建议全选，然后单击【NEXT】按钮，出现如图 3-4 所示的界面。

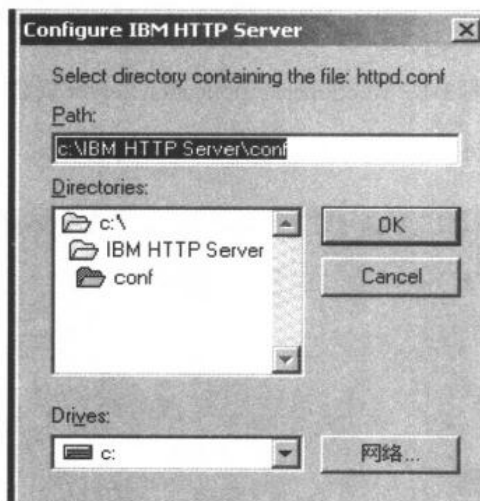


图 3-4 配置

(5) 选择 Application Server plugins 时，选择“IBM HTTP Server version 1.3.3”；配置 IBM HTTP Server，使用默认路径即可，然后单击【OK】按钮，则出现如图 3-5 所示的界面。

(6) 安装完毕，如果选中复选框，则在安装完毕后，打开 README 帮助文件。

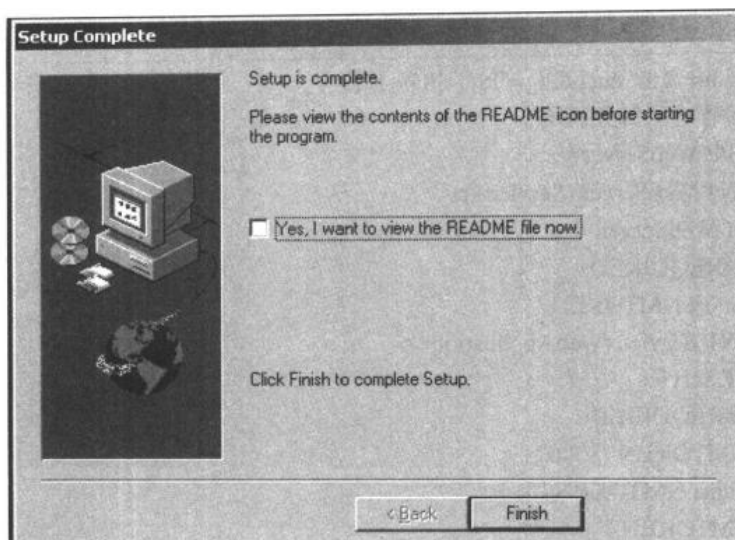


图 3-5 安装完毕

#### 4. 应用程序服务器的安装配置

##### ● Web 服务器的配置

Tomcat 的配置我们已经在上一章中讲述了, 现在我们来讲述一下 JSWDK 服务器的配置。JSWDK 服务器的配置也比较简单, 只需把软件直接拷贝到相应的目录下就可以了。在根目录下, 有个重要的配置文件 `webserver.xml`, 用浏览器查看, 如图 3-6 所示。

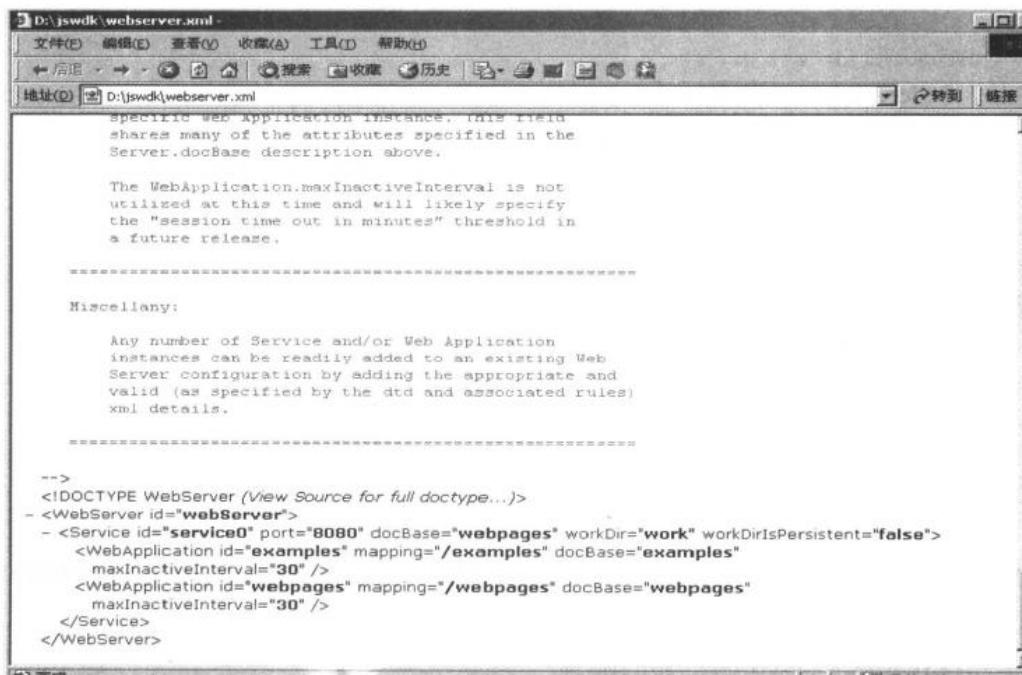


图 3-6 webserver.xml

源代码如例程 3-1 所示。



例程 3-1

```

<?xml version="1.0" encoding="ISO-8859-1"?>
.....//省略了部分代码
<!DOCTYPE WebServer [
<!ELEMENT WebServer (Service+)>
<!ATTLIST WebServer
    id ID #REQUIRED
    adminPort NMTOKEN "">
<!ELEMENT Service (WebApplication*)>
<!ATTLIST Service
    id ID #REQUIRED
    port NMTOKEN "8080"
    hostName NMTOKEN ""
    inet NMTOKEN ""
    docBase CDATA "webpages"
    workDir CDATA "work"
    workDirIsPersistent (false | true) "false">
<!ELEMENT WebApplication EMPTY>
<!ATTLIST WebApplication
    id ID #REQUIRED
    mapping CDATA #REQUIRED
    docBase CDATA #REQUIRED
    maxInactiveInterval NMTOKEN "30">
]>
<WebServer id="webServer">
    <Service id="service0">
<WebApplication id="examples" mapping="/examples" docBase="examples"/>
<WebApplication id="webpages" mapping="/webpages" docBase="webpages"/>
    </Service>
</WebServer>

```

找到如下行:

```

port NMTOKEN "8080"
    hostName NMTOKEN ""
    inet NMTOKEN ""
    docBase CDATA "webpages"
    workDir CDATA "work"
    workDirIsPersistent (false | true) "false">

```

显然, 这些语句是用来设置端口号、发布目录以及主机名的, 您可以设置成自己喜欢的名称。

#### ● JSP/SERVERLETS 服务器的配置

在 Tomcat 的设置中, 您可以创建自己的新目录。如下代码所示, 我们就创建了新目录 test 和 website。这样, 除了原有的 examples 和 root, 又多了两个目录。

```

<Context path="/examples" docBase="webapps/examples" debug="0" reloadable="true" >
    </Context>

```



```
<Context path="" docBase="webapps/ROOT" debug="0" reloadable="true" >
    </Context>
<Context path="/test" docBase="webapps/test" debug="0" reloadable="true" >
    </Context>
<Context path="/website" docBase="webapps/website" debug="0" reloadable="true" >
    </Context>
```

● JSP/SERVERLETS 服务器的启动与停止

BIN 目录下的 startup.bat 文件是 Tomcat 服务器的启动程序文件。用任何编辑器（如 Windows 自带的记事本）打开，代码如例程 3-2 所示。

例程 3-2

```
@echo off
rem $Id: startup.bat,v 1.7 2000/03/31 19:40:02 craigmcc Exp $
rem Startup batch file for tomcat server.

rem This batch file written and tested under Windows NT
rem Improvements to this file are welcome

if not "%Tomcat_HOME%" == "" goto start

SET Tomcat_HOME=.
if exist %Tomcat_HOME%\bin\tomcat.bat goto start

SET Tomcat_HOME=..
if exist %Tomcat_HOME%\bin\tomcat.bat goto start

SET Tomcat_HOME=
echo Unable to determine the value of Tomcat_HOME.
goto eof

:start
set Tomcat_HOME=d:\tomcat
set Java_home=d:\jdk1.3
call %Tomcat_HOME%\bin\tomcat start %1 %2 %3 %4 %5 %6 %7 %8 %9

:eof
```

找到行：

```
set Tomcat_HOME=d:\tomcat
set Java_home=d:\jdk1.3
```

如果您的 startup.bat 文件还没有这些行，就必须添加这些代码。否则服务器不能正常运行。

### 3.2.3 Java 集成开发环境的安装配置

下面将讲述目前最为流行的 Java 集成开发环境 JBuilder 和 Visual Age 的安装与配置。

#### 1. JBuilder 的安装

安装要求：

- Microsoft Windows 98, 2000, or NT 4.0
- Intel Pentium II 233MHz 或更高配置
- 最小内存为 128 MB
- 推荐内存为 256 MB 以上
- 硬盘空间至少为 115~250 MB

JBuilder 的安装步骤如下。

#### 步骤

(1) 在 CD 驱动器中插入 CD-ROM。如果在系统中 autorun 不能用，在 CD 驱动器的根目录下运行 install\_windows.exe。选择 FULL INSTALLATION 选项，出现如图 3-7 所示的界面。

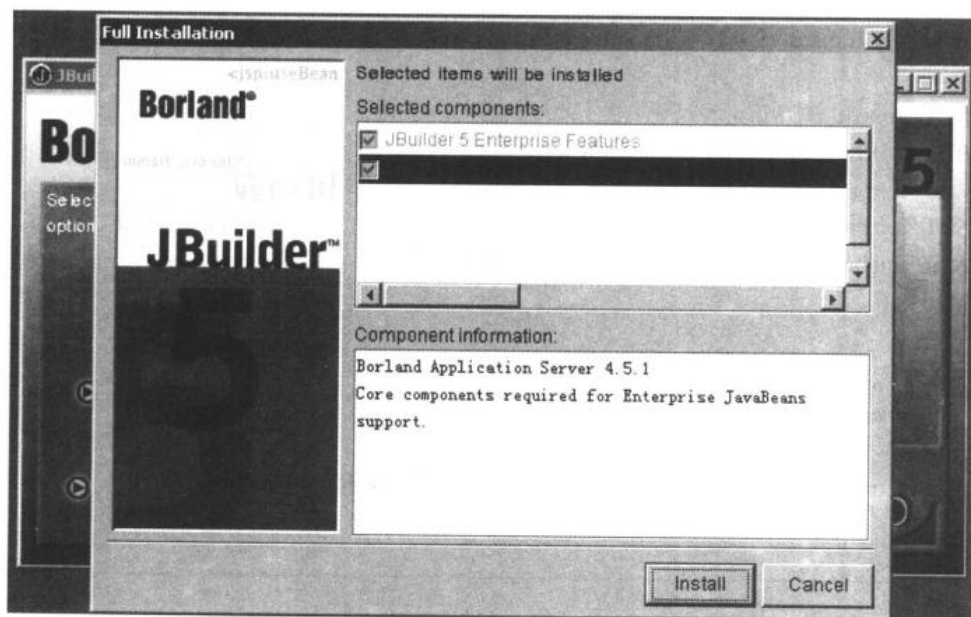


图 3-7 FULL INSTALLATION

(2) 单击【INSTALL】按钮，进行安装。然后出现协议说明对话框，选中复选框接受协议。单击【NEXT】按钮，出现如图 3-8 所示的界面。

(3) 选择安装的路径，单击【NEXT】按钮，进行安装。

(4) 安装完毕后，运行 JBuilder，并新建一工程，如图 3-9 所示。

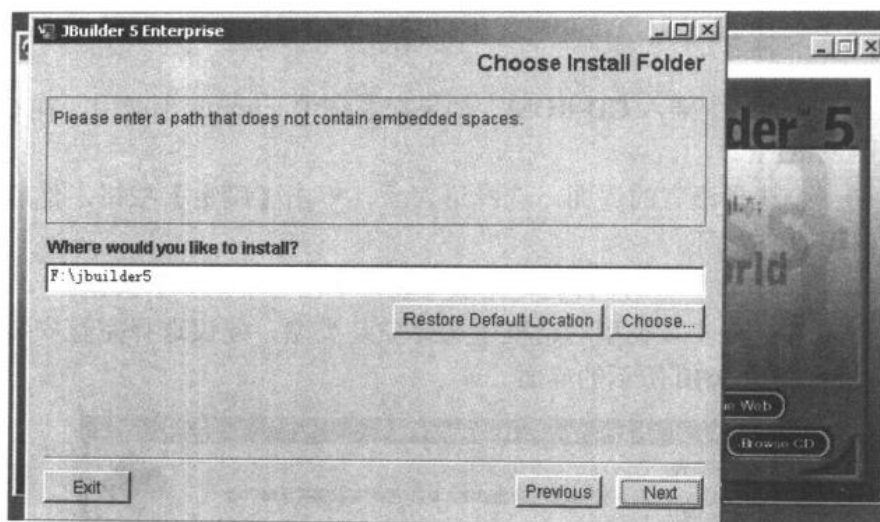


图 3-8 选择安装路径

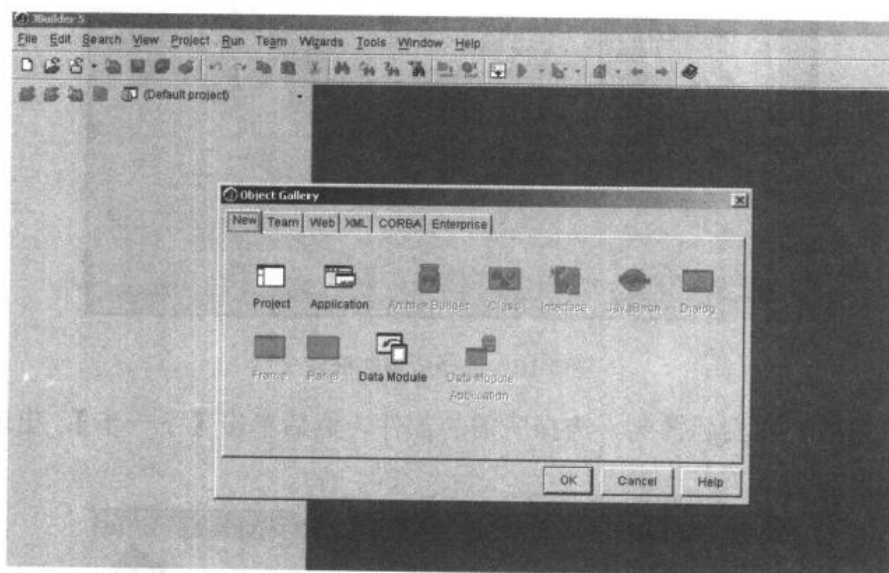


图 3-9 Jbuilder5 开发环境

## 2. Visual Age 的安装

IBM VisualAge for Java 是创建 Java 兼容的应用程序、Applet 和 JavaBean(TM) 组件的一个功能强大的快速应用程序开发工具。使用 VisualAge for Java 程序设计环境，可以创建运行在任何 Java 兼容的虚拟机 Java 开发工具箱 (JDK 1.1.6) 或在任何支持 Java 的浏览器中的 100% 的纯 Java (TM) 应用程序。

使用 VisualAge for Java 可以添加类、添加或改变方法，然后不需要退出测试环境来进行增量编译。产品包括可视化组合编辑器和一个完全集成的基于资源库的环境，提供完整的源代码和版本控制。

Visual Age 的安装步骤如下：

## 步骤

(1) 在 CD 驱动器中插入 CD-ROM。如果在系统中 autorun 不能用, 在 CD 驱动器的根目录下运行 setup.exe。

(2) 出现一个选择语言的界面, 选择语言后, 单击【确定】按钮, 然后单击【下一步】, 进入协议说明的界面。

(3) 选择接受协议的选项, 否则将无法安装, 单击【下一步】按钮。

(4) 在出现的安装类型对话框中选择安装类型, 建议选择完整安装, 单击【下一步】按钮, 进入如图 3-10 所示的界面。

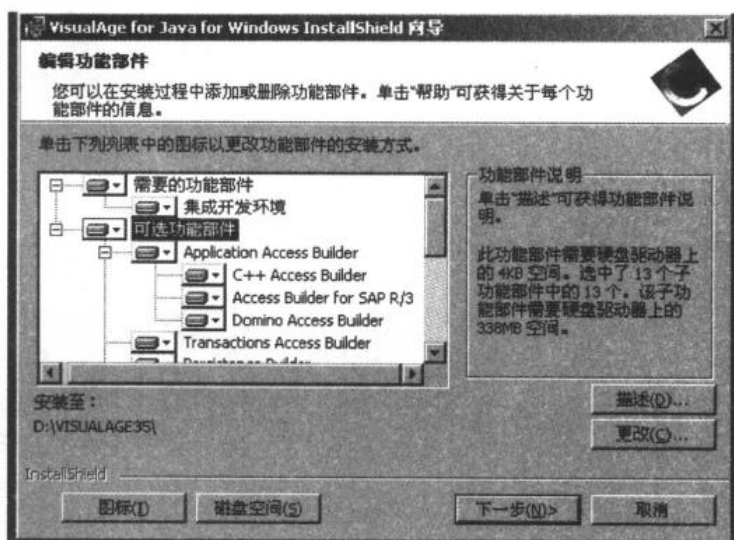


图 3-10 选择所需的部件

(5) 根据需求和实际情况, 选择所需的部件, 然后单击【下一步】, 出现如图 3-11 所示的界面。

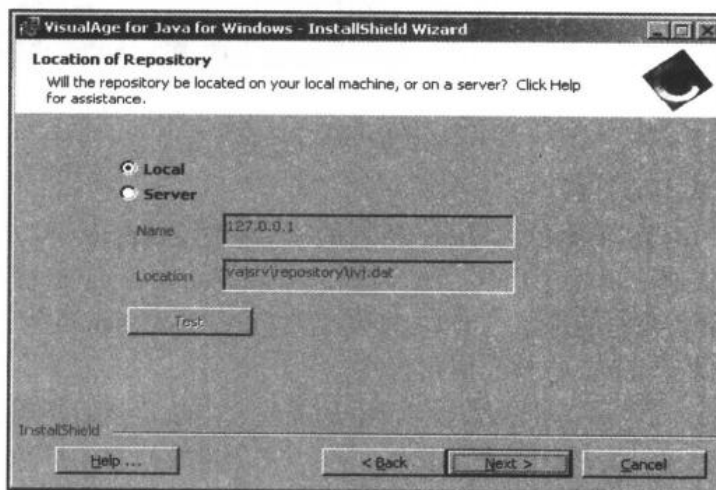


图 3-11 本地/远程安装



(6) 选择 Local 选项, 这是指在本地安装。单击【下一步】按钮, 出现如图 3-12 所示的界面。

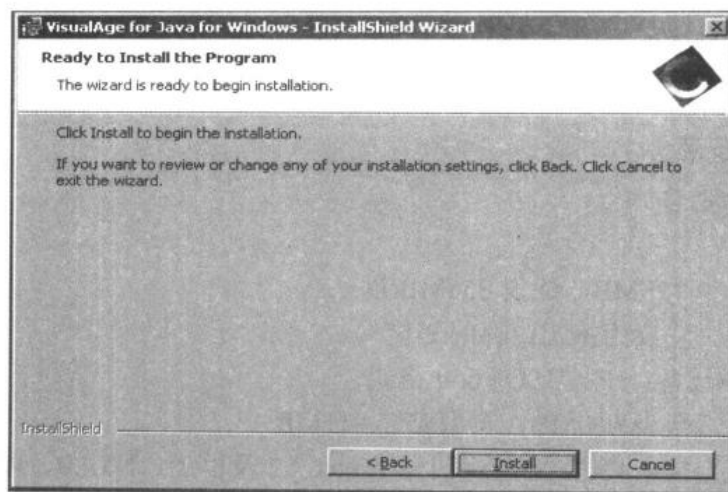


图 3-12 安装

(7) 单击【Install】按钮进行安装。

(8) 安装完毕后, 运行 Visual Age, 出现如图 3-13 所示的界面。

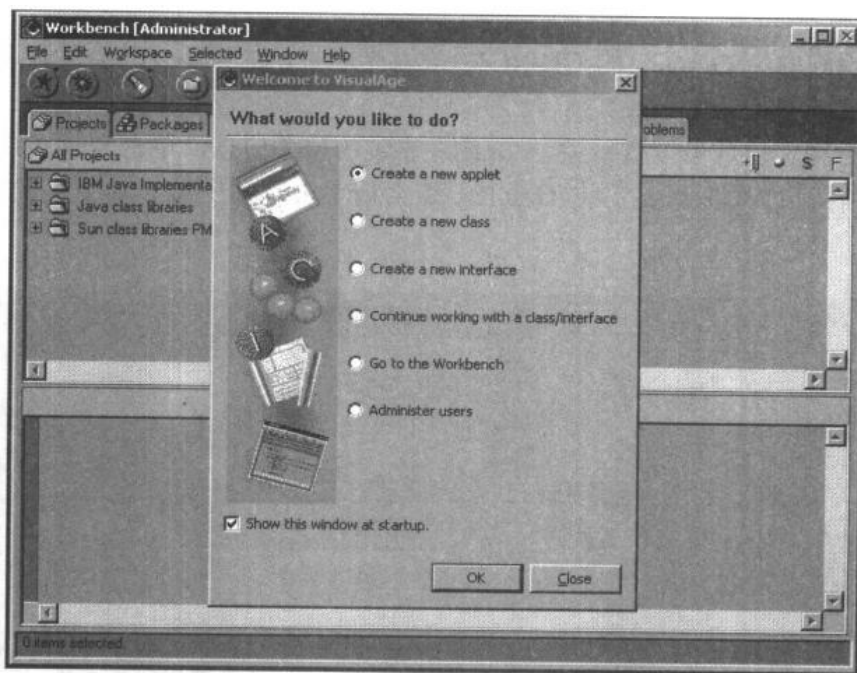


图 3-13 Visual Age 开发环境

如果您使用的 Java 集成开发环境是 JBuilder 或者是 Visual Age, 那么您就不需要任何地配置就可以编译您的 Java 文件。

在安装 JBuilder 或者是 Visual Age 时, 系统会自动地设置 PATH 和 CLASSPATH 这些环境变量。



### 3.2.4 数据库的安装配置

下面将讲述目前最为流行的数据库 Oracle 和 Microsoft SQL Server 2000 的安装与配置。

#### 3.2.4.1 数据库的安装

##### 1. Oracle 8 的安装

硬件要求：

- 内存：至少 128MB，建议 256MB 以上
- 交换空间：建议为系统内存的 2 倍
- CD-ROM 设备：支持 ISO 9660 格式
- 硬盘空间：至少 660MB，典型安装 811MB
- 操作系统：SOLARIS 2.5 及以上、Linux 2.2 及以上、NT4.0、Windows 2000

考虑到大多数读者的操作系统为 Windows 系列，下面讲述在 Windows 2000 平台下如何安装 Oracle8。

Oracle 8 的安装步骤如下：

#### 步骤

(1) 把用来安装 Oracle8 的 CD-ROM 放入光驱中，如果不支持自动启动，就在 CD-ROM 的根目录下双击 setup.exe。

(2) 单击浏览信息可以查看关于 oracle 安装的相关信息，单击查看 CD 可以浏览光盘内的所有文件。如果您想要安装 Oracle，那么请单击开始安装，则会出现如图 3-14 所示的界面。

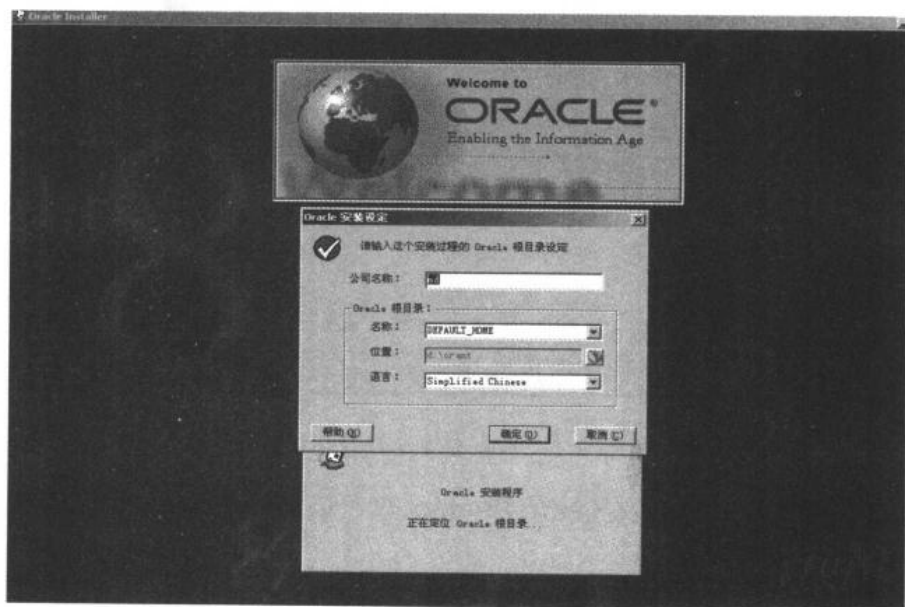


图 3-14 输入公司名称并选择语言种类

(3) 输入自己的公司名称，并选择语言种类，也可以按照默认设置。确认后，单击【确定】按钮，出现如图 3-15 所示的界面。

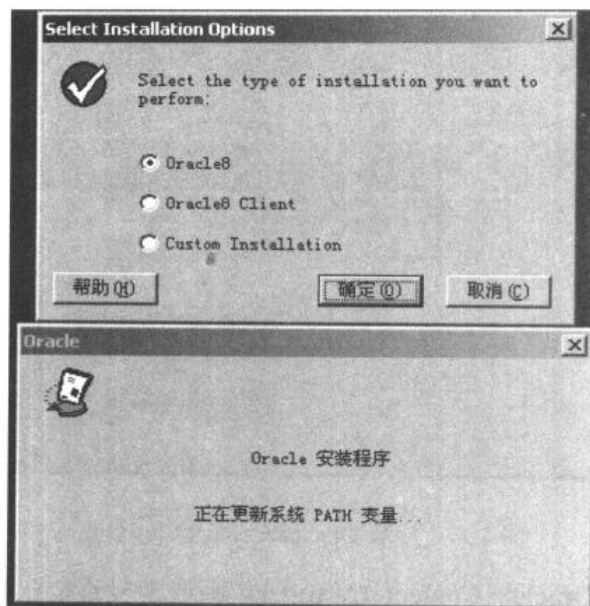


图 3-15 选择安装的类型

(4) 选择安装的类型，然后单击【确定】按钮，会出现如图 3-16 所示的界面。

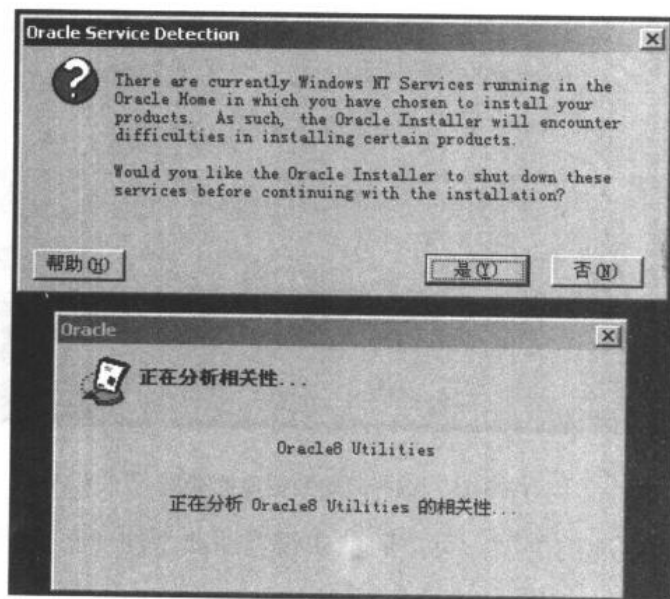


图 3-16 信息提示

(5) 在安装 Oracle 的目录下，可能正在运行着一些服务，安装过程会提示您关闭这些服务，否则不能正常安装。单击【是】按钮后，会出现如图 3-17 所示的界面。

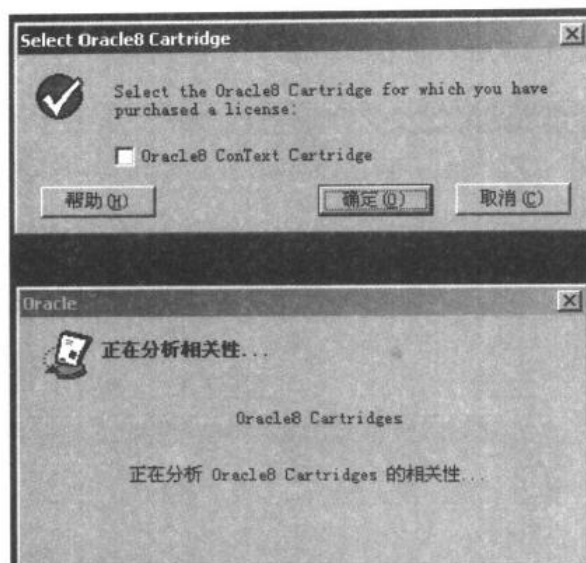


图 3-17 安装 Oracle8 Content Cartridge

(6) 如果要安装 Oracle8 Content Cartridge, 则选中复选框, 单击【确定】按钮后, 出现如图 3-18 所示的界面。

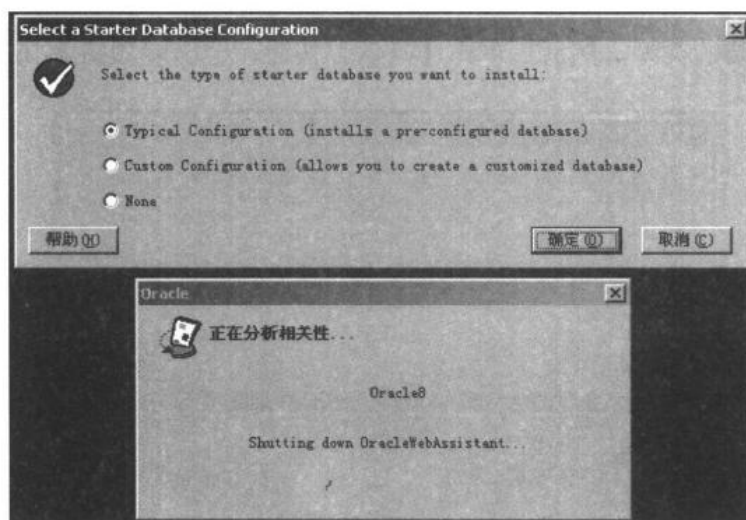


图 3-18 选择起始数据库的类型

(7) 选择起始数据库的类型, 可按照默认设置即典型注册形式进行安装。单击【确定】按钮后进行安装。

当数据库安装完毕后, 默认创建了一个数据库, 并建立了十二个用户。其中, 有三个特殊的用户:

INTERNAL  
System  
SYS

- INTERNAL (Oracle)

只能在服务器端使用的超级用户，具有 DBA 的所有特权。相当于连接到 SYS 用户，连接方式与普通用户相同。

- System (MANAGER)

在建立数据库时自动建立的一个超级用户，拥有显示管理信息的附加表和视图，以及由 Oracle 工具使用的所有表和视图。

连接方式与普通用户相同。

- SYS (CHANGE\_ON\_INSTALL)

在建立数据库时自动建立的一个超级用户，拥有显示管理信息的附加表和视图，以及由 Oracle 工具使用的所有表和视图。它拥有最高权限。

有些数据字典只能在 SYS 用户中查询，在 System 用户中查不到。

连接方式与普通用户相同。

如图 3-19 所示，既可以分别在用户名称中输入用户名，在口令处输入相应的用户口令，又可以在用户名称处直接输入用户名和口令（如图 3-20 所示）。

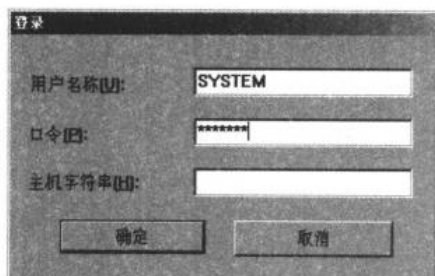


图 3-19 SQL Plus

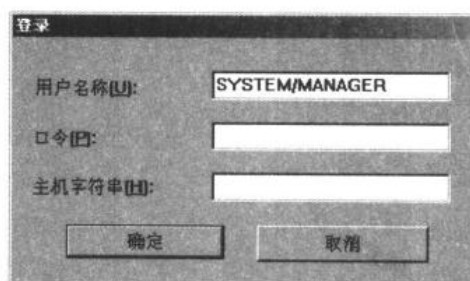


图 3-20 登录界面

登录成功后，SQL\*PLUS 窗口中就会出现 SQL>提示符。此时，用户就可以执行 SQL、PL/SQL 和 SQL\*PLUS 命令了，如图 3-21 所示。

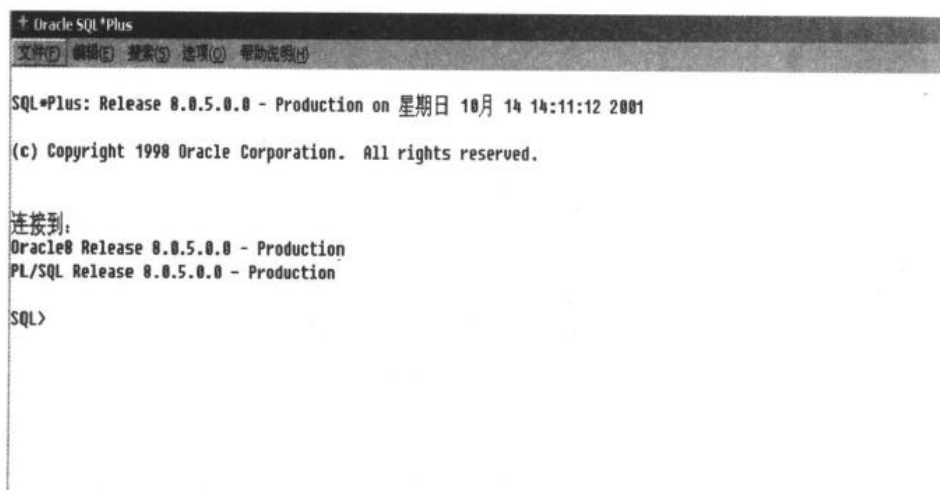


图 3-21 SQL Plus 运行界面



## 2. Microsoft SQL server 的安装

Microsoft SQL Server 2000 是在 SQL Server 7.0 版的基础上建立,并扩展了 SQL Server 7.0 版的性能、可靠性、质量以及易用性。它还包括许多新功能,这些功能进一步将 SQL Server 确立为 OLTP、数据仓储以及电子商务应用程序的最佳数据库平台。

系统要求:

- Microsoft SQL Server 2000 可在运行 Intel 或兼容的 Pentium、Pentium Pro 或 Pentium II 处理器的计算机上运行。
- 处理器必须以最低 166 MHz 的速度运行。
- SQL Server 2000 各版本的内存 (RAM) 要求如下:

企业版	最低 64 MB, 推荐使用 128 MB
标准版	最低 64 MB
个人版	在 Windows 2000 上需要 64MB, 在所有其他操作系统上需要 32MB
开发版	最低 64 MB
- Desktop Engine      在 Windows 2000 上最低需要 64 MB, 在所有其他操作系统上需要 32 MB
- 根据所选组件和安装选项, SQL Server 2000 有如下硬盘要求:

数据库组件	95~270 MB, 通常需要 250 MB
Analysis Services	最低 50MB, 通常需要 130MB
English Query	80MB
Desktop Engine	仅 44MB
- SQL Server 2000 要求显示器具有 VGA 分辨能力;
- SQL Server 图形工具要求显示器使用 800x600 或更高的分辨率。
- SQL Server 2000 需要 CD-ROM 驱动器以及 Microsoft 鼠标或兼容的指针设备。
- SQL Server 2000 需要 Internet Explorer 5.0 或更高版本, 而且在下列这些操作系统上受支持:
  - \* Windows 2000
  - \* Microsoft Windows NT 4.0 版 Service Pack 5 或更高版本
  - \* Windows Millennium Edition
  - \* Windows 98
  - \* Windows 95 (仅客户端连接选项)

在 Windows 95 上安装 SQL Server 2000 软件之前,必须安装用于 Windows 95 的 Winsock 2 Update。SQL Server 2000 光盘上提供此更新程序,并可从 SQL Server 2000 光盘所显示的自动运行窗口中,选择“SQL Server 2000 前提条件”选项来安装它。

Windows NT 4.0 终端服务器上不支持 SQL Server 2000。

### 3.2.4.2 数据库的配置

下面我们以前最为流行的数据库 Oracle 为例来讲述一下数据库的配置。

安装完 Oracle 后,需要配置 Oracle 数据库示例模式与数据。由于在配置过程中使用了 SQL Plus,因此需要在机器上建立一个 Oracle 网络客户。可以按照以下的方法来建立

一个 SQL 网络用户:

### 步骤

(1) 启动 Network Administration (网络管理员) 的 Net8 Easy Config 应用程序以配置 SQL 的网络用户。

(2) 在系统询问时, 确保创建了一个名为 WXN 的新网络服务名。

(3) 在系统询问时, 选择 TCP/IP 作为自己使用的协议。

(4) 在系统询问时, 输入机器的主机名 (例如本地机可以是 Local Host), 然后输入端口号 1521 (默认端口号)。

(5) 在系统询问时, 输入服务名 WXN。

在配置完 SQL 网络客户之后, 就可以使用 SQL PLUS 来创建数据库并且加载数据。在系统询问时, 可以输入 system 作为用户名, manager 作为口令, wxn 作为主机字符串。

在 Oracle 的配置属性文件中, 可以使用 JDBC:Oracle:THIN:@LOCAL HOST:1521:WXN 来配置这个属性。如果是在远程机器上安装了数据库, 那么就必须将这个属性设置中的 Local Host 值用远程机器的主机名或者 IP 地址来替代。

## 3.3 Tomcat UNIX/Windows 服务器安装与配置

### 3.3.1 Tomcat 的安装

#### 3.3.1.1 UNIX 下 Tomcat 的安装:

##### 1. 软件下载

到下列网址去下载:

<http://jakarta.apache.org/builds/tomcat/release/v3.1/> 下载 jakarta-tomcat.tar.gz

<http://jakarta.apache.org/builds/tomcat/release/v3.1/bin/linux/i386/> 下载 mod\_jserv.so

这是在 UNIX 系统下支持 JSP 的模块, 作用等同于 Windows 系统下的 DLL(DYNAMIC LINKED LIBRARY 动态链接库) 文件。

##### 2. 软件安装

- 在 UNIX 系统下安装 Tomcat

```
#cp Jakarta-tomcat.tar.gz/usr/local
```

```
#tar xvzf Jakarta-tomcat.tar.gz
```

执行上述操作之后, 退出系统并重启机器。

- 在 UNIX 系统下运行 Tomcat

```
#./startup.sh start
```

```
#lynx http://localhost:9090/
```

如果出现了 JSP/Servlet 例子的页面显示, 并且 JSP/Servlet 的例子也能够运行, 那么



说明您已经成功安装了 Tomcat 服务器。

如果您想关闭 Tomcat，那么您只要执行下述命令即可。

```
#./shutdown.sh stop
```

执行了上述命令，您就已经结束了 Tomcat 服务器的服务。这时，如果试着去运行 JSP/Servlet，则会出现不能正确显示的页面说明。

### 3.3.1.2 Windows 下 Tomcat 的安装

#### 1. 软件下载

如果您还没有 Tomcat 软件，那请您到下面的网址去下载：

<http://jakarta.apache.org/builds/tomcat/release/v3.1/>

下载得到的文件应该是一个压缩文件，名字叫 Tomcat.ZIP。

#### 2. 软件安装

用 WinZIP 或其他解压缩软件把 tomcat.zip 解压缩到一个目录下（例如把它解压缩到 D 盘）。这样，系统会自动创建 tomcat 子目录。于是，在您的计算机系统的 D 盘中，将会有有一个叫做 Tomcat 的目录。

可见，在 Windows 系统下，安装 Tomcat 非常简单。只要解压缩就行，不需要其他任何程序的运行。

至此，在 UNIX 和 Windows 两种操作系统下，安装 Tomcat 的步骤已经讲完了。

下面，我们来讲述一下 Tomcat 在 UNIX 和 Windows 两种系统下的配置。

### 3.3.2 Tomcat 的配置

在 D:\Tomcat 目录下，我们看到一个 CONF 文件夹，在这个文件下有一个文件，名字是 SERVER.XML。打开这个文件，将会看到下面的一段代码：

```
<Connector className="org.apache.tomcat.service.SimpleTcpConnector">
  <Parameter name="handler" value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
  <Parameter name="port" value="9090"/>
</Connector>
```

Tomcat 可以作为一个独立的服务器使用，所以 Tomcat 有自己的端口号。这个端口号可以修改，建议设置为您的机器上还没有被占用的端口号。例如，我在我的机器上设置了 Tomcat 服务器的端口号为 9090。

为了使得 Tomcat 服务器能够很好的运行，需要设置如下的环境变量。

#### 1. 在 Windows 98 系统下的设置

找到 AUTOEXEC.BAT 文件，打开该文件，在相应的位置输入如下代码：

```
PATH=%PATH%;D:\jdk1.3
set CLASSPATH=D:\jdk1.3\lib\Tools.jar;D:\jdk1.3\lib\dt.jar
set Java_HOME=D:\jdk1.3
```

在上述设置中，JDK 和 Tomcat 均被安装在 D 盘下。若是安装在其他的盘下，上述代码只需做相应的改变即可。

## 2. 在 Windows 2000/Windows NT 下的设置

## 步骤

(1) 用右键单击“我的电脑”图标，出现如图 3-22 所示的菜单。



图 3-22 设置环境变量

(2) 在菜单的最下方，有一“属性”选项，出现“系统特性”对话框，然后再单击对话框中的“高级”选项卡，则出现如图 3-23 所示的界面。

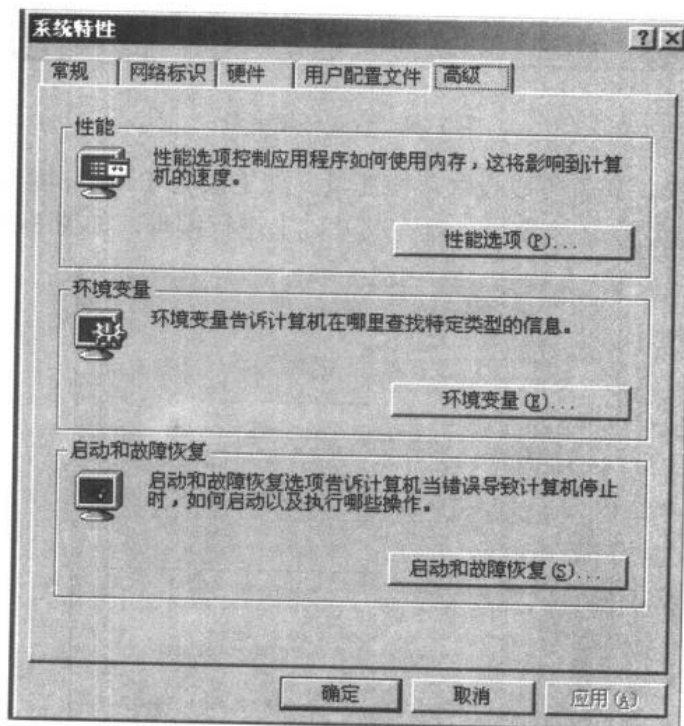


图 3-23 设置环境变量

(3) 在这个界面上单击【环境变量】按钮，出现如图 3-24 所示的界面。

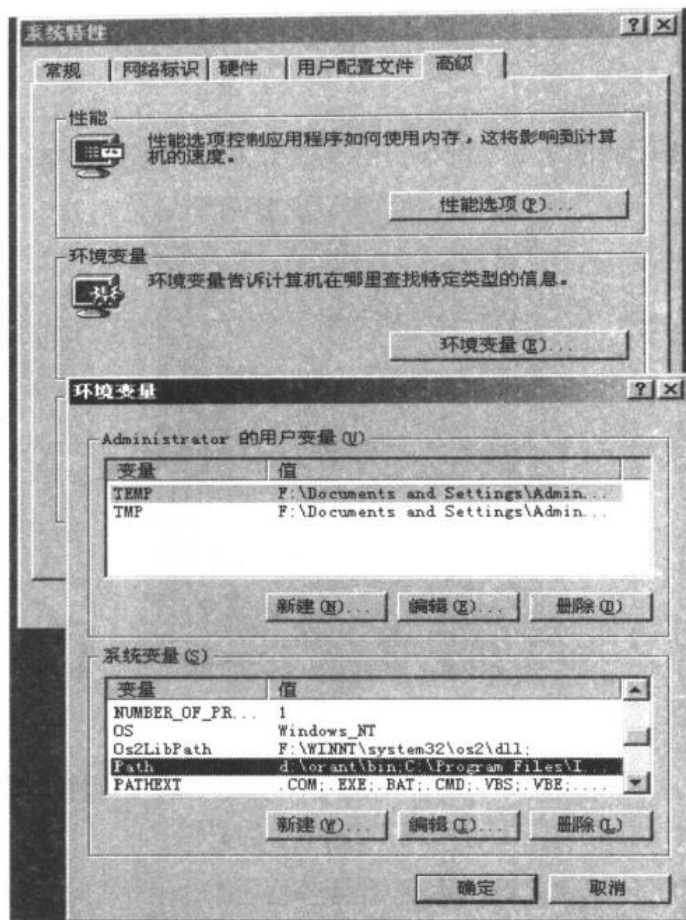


图 3-24 设置环境变量

(4) 这个页面中，您就可以设置环境变量，如果没有 Path 环境变量，则新建这个环境变量。如果有 Path 这个环境变量，就在其基础上进行编辑，如图 3-25 所示。



图 3-25 设置 Path 变量

(5) 在“变量值 (V)”内输入 d:\jdk1.3, 输入完毕后, 单击【确认】按钮进行确认。  
同样, 如果有 Java\_HOME 这个环境变量, 就其基础上进行编辑, 如图 3-26 所示。  
如果没有 Java\_HOME 这个环境变量, 则新建这个环境变量。

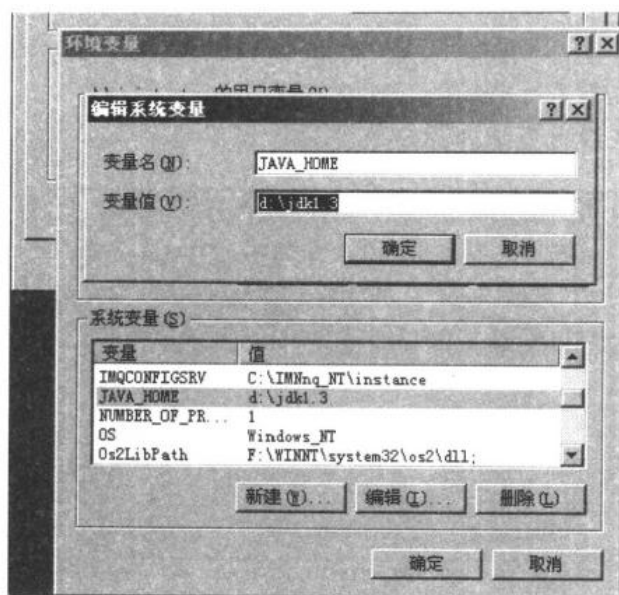


图 3-26 设置 Java\_HOME 变量

(6) 在“变量值 (V)”内输入 d:\jdk1.3, 输入完毕后, 单击【确认】按钮进行确认。  
同样, 如果有 CLASSPATH 这个环境变量, 就其基础上进行编辑, 如图 3-27 所示。  
如果没有 CLASSPATH 这个环境变量, 则新建这个环境变量。

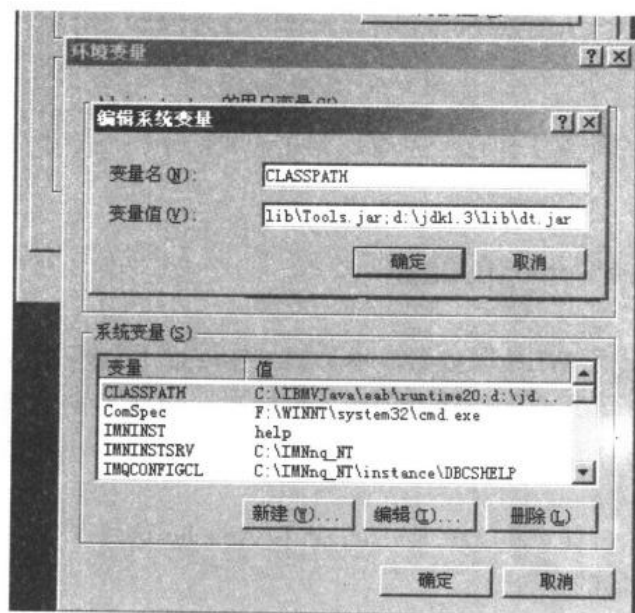


图 3-27 设置 CLASSPATH 变量



(7) 在“变量值(V)”内输入 `d:\jdk1.3\lib\Tools.jar;d:\jdk1.3\lib\dt.jar`; 输入完毕后, 单击【确认】按钮进行确认。

至此, Tomcat 的环境变量配置已经讲完了。

现在启动 BIN 下的 `startup.bat` 文件, 在执行时可能会出现“Out of Environment Space”字样的错误。出现错误的原因是您的 DOS 窗口的缓冲区设置得太小, 解决的方法就是将 DOS 窗口的缓冲区设置得大一些。

DOS 窗口的缓冲区设置方法如图 3-28 所示。

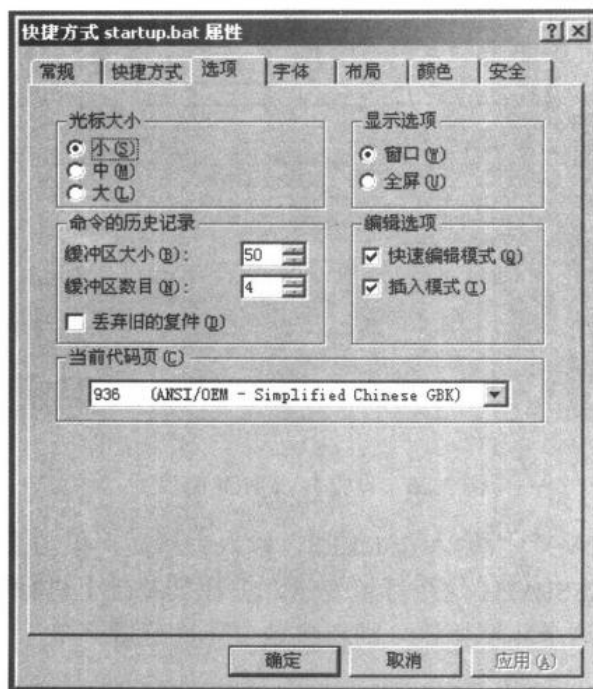


图 3-28 缓冲区设置

到现在, Tomcat 的配置工作已经完成。下面, 我们来讲述一下 Tomcat 与 Web Server 在实际中的配合使用。

### 3.3.3 Tomcat 和 Web Server 的配合

Tomcat 本身就是一个 WEB 服务器, 本身就可以运行 JSP 和 Servlet。当然, 我们也可以把 Tomcat 与 Apache 等其他服务器结合在一起使用, 并且 Tomcat 服务器和 Apache 服务器两者可以不在同一台计算机上, 也就是说, 可以在两台计算机上分别安装这两个服务器软件。

下面我们以在同一台计算机上安装 Tomcat 和 Apache 这两个软件为例, 来讲解 Tomcat 与 Apache 等其他服务器的配合。

#### 1. 在 UNIX 系统下

```
#cp mod_jserv.so/apache/libexec
#cp /usr/local/Jakarta-tomcat/conf/tomcat.conf/apache/conf/path
```

```
#vi /apache/conf/path/httpd.conf
```

加入下面的语句:

```
include /apache/conf/path/httpd/tomcat.conf
```

其作用是使 Apache 服务器在启动时加载 Tomcat 模块。

## 2. 在 Windows 系统下

### ● 下载软件 (Apache Web Server 和 Jserv)

首先到<http://www.apache.org>/网站上去下载一个 Apache Web Server。

然后再到<http://Java.apache.org> 站点上去下载一个 Jserv。

### ● 安装软件

先安装 Apache Web Server, 例如: 安装在 D:\Apache 目录下。

安装 Jserv 需要指定 JVM 所在的目录, 例如: D:\Tomcat\LIB\。

### ● 配置

安装完上述软件后, 您可以修改文件 D:\Apache\CONF\HTTPD.CONF。在这个文件中, 您可以设置自己喜欢的站点名 (SERVER NAME), 也可以重新设置自己喜欢的端口号 (PORT)。

需要注意的是您需要修改文件 D:\Apache\CONF\HTTPD.CONF, 以便支持 JSP 与 Servlet。在这个文件的最后, 您需要加上 INCLUDE Tomcat\_HOME\CONF\ Tomcat-Apache.CONF。

## 3.3.4 在 Tomcat 中建立新的 Web 应用程序

### 1. 在 Tomcat 中建立一个新的 JSP 页面进行测试

basic1.jsp 文件的源代码如例程 3-3 所示。

例程 3-3

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>测试一.jsp</title>
</head>
<body>
<div align="center">
<%
int i;
for(i=1;i<=10;i++)
{
  out.println("<font size="+i+">"+i+"</font><br>");
}
%>
</div>
</body>
</html>
```



将上述代码存为 D:\TOMCATWEBAPPS\TEST\BASIC\BASIC1.JSP，启动服务后，在浏览器的地址栏内输入 `http://localhost:8080/jspbook/chapter3/basic1.jsp`。运行结果如图 3-29 所示。

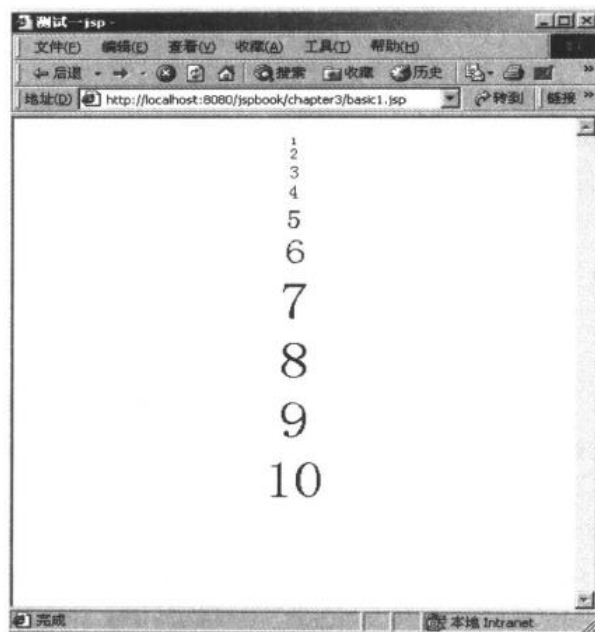


图 3-29 BASIC1.JSP 运行结果

## 2. 在 Tomcat 中建立一个新的 Servlet 进行测试

`basicServlet.Java` 文件的源代码如例程 3-4 所示。

例程 3-4

```
package basic2;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class basicServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化全局变量*/
    public void init() throws ServletException {
    }
    /**响应 HTTP 的 GET 方法*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
```

```
out.println("<title>测试二 servlet</title>");
out.println("</head>");
out.println("<body>");
out.println("<div align=center>");
int i;
for(i=10;i>=1;i--)
{
    out.println("<font size="+i+">"+i+"</font><br>");
}
out.println("</div>");
out.println("</body>");
out.println("</html>");
}
```

将上述代码存为 D:\TOMCAT\WEBAPPS\TEST\ WEB-INF\CLASSES\BASIC2\ basic Servlet.Java。用 JavaC 进行编译后启动服务，然后在浏览器的地址栏内输入 <http://localhost:8080/jspbook/servlet/basic2.basicServlet>。运行结果如图 3-30 所示。

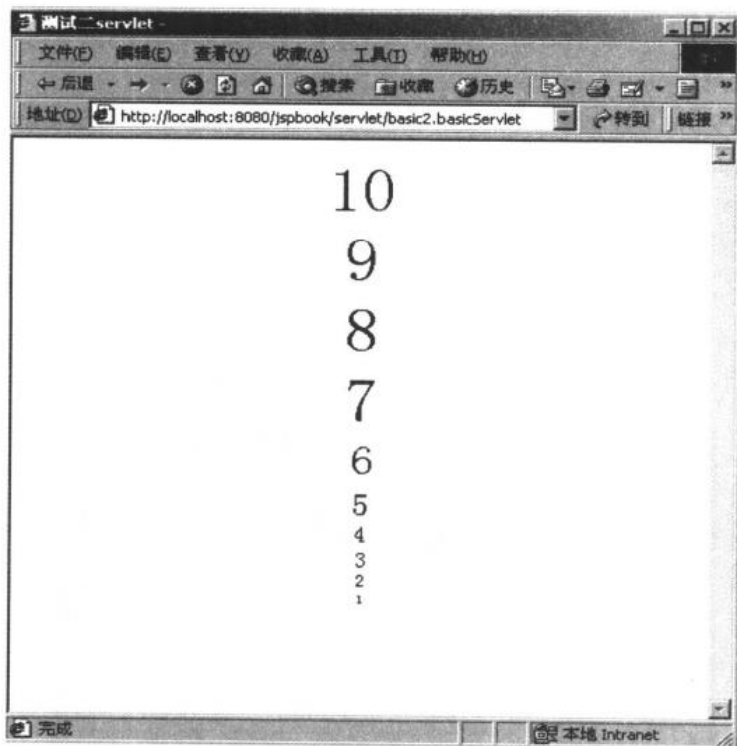


图 3-30 /servlet/basic2.basicServlet 运行结果



需要注意的是之所以在输入地址时，同时要输入“basic2.”，其原因是因为上述的代码中有程序包语句“package basic2;”。如果不输入“basic2.”，系统则会报错。

## 3.4 安装配置中的常见问题与解决方法

### 1. 配置 Oracle 的 JDBC 的具体步骤

- 数据库与 Web 应用服务器属于同一机器的情况

操作系统为 Win98、Windows NT、Windows2000 系列。

解决方法：

增加下面的 ClassPath 环境变量（[Oracle\_Home]表示 Oracle 安装的目录）：

[Oracle\_Home]\jdbc\lib\classes12.zip

[Oracle\_Home]\jdbc\lib\nls\_charset12.zip

· 增加下面的 PATH 环境变量：

[Oracle\_Home]\jdbc\lib

- 数据库与 Web 应用服务器不属于同一机器的情况

操作系统为 Win98、Windows NT、Windows2000 系列。

解决方法：

(1) copy 文件 classes12.zip、nls\_charset12.zip 到任何目录，如 c:\jdbc\_driver；

(2) 增加下面的 ClassPath 环境变量（[Oracle\_Home]表示 Oracle 安装的目录）：

c:\jdbc\_driver\classes12.zip

c:\jdbc\_driver\nls\_charset12.zip

增加下面的 Path 环境变量：

c:\jdbc\_driver

### 2. 中文操作系统下的 JDK 安装

在安装 JDK 时，如果操作系统为中文系统，那么系统注册表会出现问题。

解决的方法为：

(1) 首先打开注册表，例如用 regedit 打开，如图 3-31 所示。

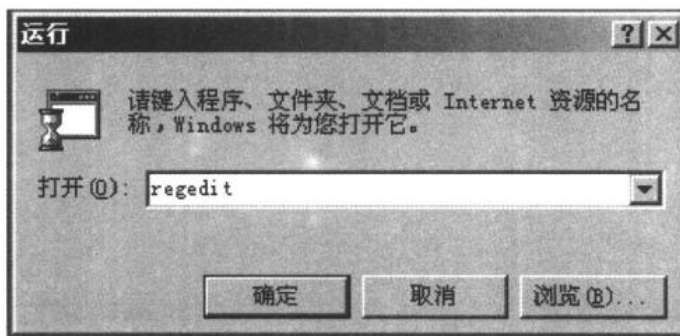


图 3-31 打开注册表

(2) 在注册表中查找 Javasoftware，位置为 HKEY\_LOCAL\_MACHINE--->SOFTWARE--->JavaSoft，在其中找到“Java 运行时环境”文件夹，将其修改为“Java Runtime Environment”，如图 3-32 所示。

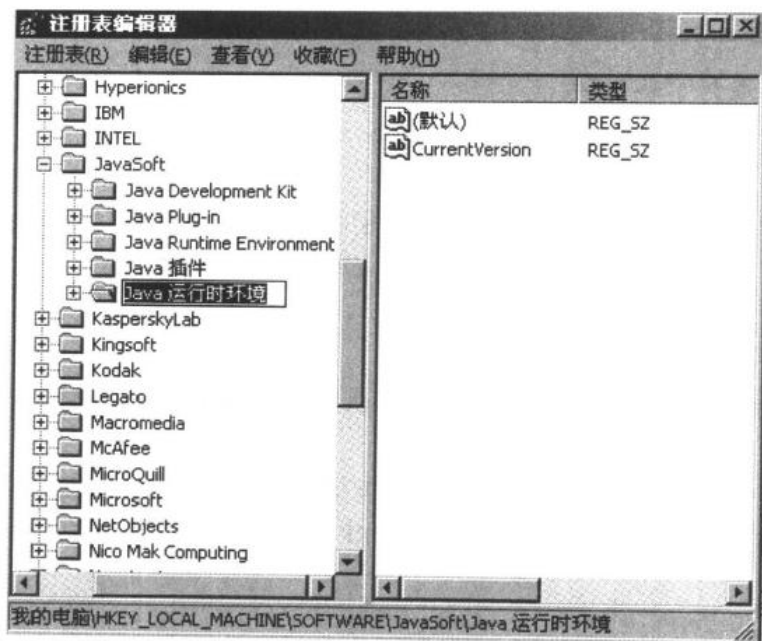


图 3-32 修改注册表

### 3. 使用 Oracle Thin JDBC Driver 可能出现的问题

- “error , connect error , No suitable driver”

解决方法:

检查数据库的 URL 中的 “.” 是否是被误写为 “;”。

- “error , connect error , IO exception: The Network Adapter could not establish the connection”

解决方法:

检查数据库的 URL 中 IP 地址以及端口号是否正确。

- “error , connect error , IO exception: Invalid connection string format , a valid format is : “ host: port :sid” ”

解决方法:

检查数据库的 URL 中 IP 地址以及端口号是否正确。

- “error , connect error , Invalid Oracle URL specified : OracleDriver connect”

解决方法:

检查数据库的 URL 中 “@” 之前是否漏写 “:”。

- “Refused: OR=(CODE=12505)(EMFI=4)”

解决方法:

检查数据库的 URL 中的 “SID” 是否正确。

- “String Index out of range: -1”

解决方法:

检查数据库的服务器地址、端口号以及数据库 SID 是否正确。

- “Database error 1, connect ,..... Protocol violation,0,null”

解决方法:

JDBC 驱动与数据库不兼容。

- “error , driver connect error , oracle.jdbc.driver.OracleDriver”

解决方法:

检查驱动是否存在, 或者驱动以及存在, 但没有加入 CLASSPATH 环境变量中。

- “error , driver connect error , oracle/jdbc/driver/OracleDriver”

解决方法:

检查数据库的 URL 是否完全正确。

上述几种情况, 笔者在实际的应用中经常遇到。在这里列举出其解决方法以供读者参考。

在下一篇中, 将会讲述 JSP 的核心内容, 包括 JSP 语法的详细讲述及其实例分析、JSP 内建对象的讲述及其实例分析、JavaBeans 在 JSP 中的应用以及在 JSP 中的文件操作。



# 开发专家之

# Sun ONE

## 第二篇 JSP 核心内容

在本篇中，将讲述 JSP 应用开发中的核心内容。

本篇分为 4 章，分别讲述了 JSP 语法及其实例分析、JSP 内建对象及其实例分析、JavaBeans 在 JSP 中的应用以及在 JSP 中的文件操作。

在第 4 章中，首先讲述了 JSP 程序的基本结构，接着又讲述了 JSP 的 Page 指令、Include 指令，以及 JSP 的动作指令。最后一节是一个实例。在第 5 章中，首先讲述了 JSP 内建对象的基本概念，接着又讲述了 JSP 的内建对象，最后一节为一个实例。在第 6 章中，首先讲述了 JavaBeans 的基本概念，接着讲述了 JavaBeans 的属性和方法，最后一节通过实例讲述了 JavaBeans 在 JSP 中的应用。第 7 章讲述了 JSP 从文件中读取数据、写入数据及向文件中追加数据的操作，最后一节讲述了一个构建计数器的实例。

通过对本篇的学习，将会掌握 JSP 技术的核心内容以及在实际开发中的应用。





## 第4章 JSP 语法详解

在本章中，将会讲述 JSP 的详细语法及其实例分析。首先讲述 JSP 程序的基本结构，接着又讲述了 JSP 的 Page 指令、Include 指令，在第 3 节中则是讲述了 JSP 的动作指令，包括 Include 指令、Forward 指令、UseBean 指令、Getproperty 指令、Setproperty 指令和 Plugin 指令。

通过对本章的学习，将会掌握 JSP 的详细语法及开发中应用是如何得到的。

### 4.1 JSP 程序的基本结构

JSP 代码一般情况下是由普通的 HTML 语句和特殊的嵌入标记组成的。可以使用任何的编辑工具并按照常规方式来书写 HTML 语句，然后将动态部分用特殊的标记嵌入即可，这些标记通常以“<%”开始并以“%>”结束。

例如：

```
<html>
<head><title> JSP 程序的基本结构</title></head>
<body>
<%
String str1;
String str2;
str1="JSP 程序的基本结构";
str2="您懂了吗";
out.println(str1);
%>
<%out.println(str2);%>
</body>
</html>
```

它将输出：

```
JSP 程序的基本结构
您懂了吗
```

构造一个 JSP 页面，除了可内嵌的规则 HTML 语言外，还有三类主要的 JSP 元素：

**Scripting elements：**可以定义最终转换为 Servlet 的部分。

**Directives：**可以控制这个 Servlet 的整体结构。

**Actions：**可以指定可重用的已有组件；另外，还可以控制 JSP 引擎的运行。

JSP 页面可能还包含注释、声明、表达式和程序段。

#### 1. 注释

- **HTML 注释：**在客户端显示一个注释。

JSP 语法:

```
<!-- comment [ <%= expression %> ] -->
```

例如:

```
<!--这是一个例子 1 -->
```

在客户端的 HTML 源代码中产生和上面一样的数据。

```
<!--这是一个例子 1 -->
```

例如:

```
<!--当前时间为: <%= (new java.util.Date()).toLocaleString() %> -->
```

在客户端的 HTML 源代码中显示为:

```
<!--当前时间为: 2001-10-5 0:09:53-->
```

描述:

这种注释和 HTML 中很像, 惟一不同之处就是, 可以在这个注释中用表达式 (例子 2 所示)。这个表达式是不定的, 由页面来决定。

- 隐藏注释: 写在 JSP 程序中, 但不发给客户。

JSP 语法:

```
<%-- comment --%>
```

例如:

```
<%@ page language="java" %>
<html>
<head><title>隐藏注释</title></head>
<body>
<font color="red">隐藏注释</font>
<%--隐藏注释隐藏注释隐藏注释隐藏注释隐藏注释--%>
</body>
</html>
```

描述:

用隐藏注释标记的字符会在 JSP 编译时被忽略。

JSP 编译器不会对<%--and--%>之间的语句进行编译, 它不会显示在客户的浏览器中, 也不会源代码中看到。

## 2. 声明

在 JSP 程序中声明合法的变量和方法。

JSP 语法:

```
<%! declaration; [ declaration; ] ... %>
```

例子:

```
<%! String name = "wxn"; %>
<%! boolean a,b,c; %>
<%! Func a = new Func(param1,param2,param3); %>
```

描述:

声明将要在 JSP 程序中用到的变量和方法。可以一次性声明多个变量和方法, 只要以“;”结尾就行。

## 3. 表达式

包含一个符合 JSP 语法的表达式。

JSP 语法:

```
<%= expression %>
```

例如:

```
<%
ResultSet RSmem = workM.executeQuery(strSQLmem);
while(RSmem.next()){
    t_member=t_member+1;
}
%>
```

描述:

表达式元素表示的是一个在脚本语言中被定义的表达式, 在运行后被自动转化为字符串, 然后插入到这个表达式在 JSP 文件的位置中并显示。

#### 4. Scriptlet (包含一个有效的程序段)

JSP 语法:

```
<% code fragment %>
```

例如:

```
<%      int t_member,t_article,t_faq;
        t_member=0;
        t_article=0;
        t_faq=0;
        String strSQLmem="SELECT id FROM member";
        String strSQLart="SELECT id FROM article";
        String strSQLfaq="SELECT id FROM faqs";
        ResultSet RSmem = workM.executeQuery(strSQLmem);
        while(RSmem.next()){
            t_member=t_member+1;
        }
        ResultSet RSart = workM.executeQuery(strSQLart);
        while(RSart.next()){
            t_article=t_article+1;
        }
        ResultSet RSfaq = workM.executeQuery(strSQLfaq);
        while(RSfaq.next()){
            t_faq=t_faq+1;
        }
%>
```

描述:

一个 Scriptlet 能够包含多个 JSP 语句、方法、变量以及表达式。

## 4.2 JSP 指令

下面我们来详细介绍 JSP 的指令。

### 4.2.1 Page 指令

Page 指令用来定义 JSP 文件中的全局属性。

在一个 JSP 页面中，可以定义一个或多个 Page 指令。但是，除了 import 属性以外的其他几种属性，只能定义一次。也就是说，如果在一个 JSP 页面中定义了多个 Page 指令，那么只有第一个 Page 指令是有效的，接下来定义的 Page 指令则是无效的。

您可以定义多个包含 import 属性的 Page 指令。例如：

```
<% page attribute="value"...%>
```

其中，attribute= language| import| contentType| session| buffer| autoFlush| isThreadSafe| info| errorPage| isErrorPage| extends  
value='...'|"..." (单引号|双引号字符)

其语法为：

```
<%@ page
    [ language="java" ]
    [ import="{package. class | package.*}, ..." ]
    [ contentType ="TYPE; charset=CHARSET" ]
    [ session="True | false" ]
    [ buffer="none | 8kb | sizekb" ]
    [ autoFlush="True | false" ]
    [ isThreadSafe="True | false" ]
    [ info="text" ]
    [ errorPage="relativeURL" ]
    [ isErrorPage="True | false" ]
    [ extends="package. class" ]
%>
```

例如：

```
<%@ page import="java.io.*, java.lang.*, java.util.Vector" %>
<%@ page language ="java" %>
<%@ page contentType="mimeType; charset=characterSet" | "text/html ; charset=ISO-8859-1"%>
<%@ page buffer="12kb" autoFlush="True" %>
<%@ page errorPage="error.jsp" %>
```

描述：

<%@ page %>指令作用于整个 JSP 页面，同样包括静态的包含文件。但是<% @ page %>指令不能作用于动态的包含文件，无论您把<% @ page %>指令放在 JSP 的文件的哪位置，它的作用范围都是整个 JSP 页面。不过，为了 JSP 程序的可读性，以及好的编程习惯，最好还是把它放在 JSP 文件的顶部。

属性：

1. language="java"

声明脚本语言的种类，默认情况下为“java”。可能的其他值为 java, javascript 和 web1。

例如：

```
<% page language="java"%>
```



```
<% page language="javascript"%>
```

```
<% page language="web1"%>
```

## 2. import="{package.class | package.\* }, ..."

需要导入的 Java 包的列表，这些包就作用于程序段、表达式以及声明。

下面的包在 JSP 编译时已经导入了，所以您就不需要再指明了。

```
java.lang.*
```

```
javax.servlet.*
```

```
javax.servlet.jsp.*
```

```
javax.servlet.http.*
```

例如：

```
<% page import="java.io.*,java.util.Hashtable" %>
```

## 3. contentType = "TYPE; charset=CHARSET"

设置 MIME 类型。默认 MIME 类型是 text/html，默认字符集为 ISO-8859-1。

语法如下：

```
<% page contentType = "TYPE; charset=CHARSET" %>
```

例如：

```
<% page contentType = "text/html; charset=ISO-8859-1" %>
```

```
<% page contentType = "text/plain" %>
```

可以查看下列地址来了解更多的 MIME 类型：

<ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types>

可以查看下列地址来了解更多的字符集：

<http://java.sun.com/products/jdk/1.1/docs/guide/intl/encoding.doc.html>

## 4. session="True | false"

设置客户是否需要 HTTP Session。

如果它为 True，那么 Session 是有用的。

如果它为 false，那么您就不能使用 session 对象，以及定义了 scope=session 的 <jsp:useBean> 元素。这样的使用会导致错误。

默认值是 True。

## 5. buffer="none | 8kb | sizekb"

buffer 的大小被 out 对象用于缓存处理执行后的 JSP 对客户浏览器的输出。

“none”是指没有任何缓存，直接输出到客户端浏览器。

您可以通过制定 buffer 的大小来制定缓存处理的大小。

默认值是 8KB。

## 6. autoFlush="True | False"

用来设置当 buffer 溢出时，是否需要强制输出。如果其值被定义为 True，则输出正常；如果它被设置为 false，当 buffer 溢出时，就会导致一些意外错误的发生。如果把 buffer 设置为 none，那么就不能把 autoFlush 设置为 false。

默认值是 True。

## 7. isThreadSafe="True | false"

用来设置 Jsp 文件是否能多线程使用。

如果设置为 True，那么一个 JSP 能够同时处理多个用户的请求；相反，如果设置为

false, 一个 JSP 只能一次处理一个请求。

默认值是 True。

8. info="text"

在 JSP 被执行时, 用来描述当前 JSP 文件的相关信息。您可以通过使用 Servlet.getServletInfo()方法取得。

9. errorPage="relativeURL"

设置处理异常事件的 JSP 文件。



如果您设置 autoFlush 为 True, 当该页的输出产生意外时, 可能会导致失败。

10. isErrorPage="True | false"

设置此页是否为出错页。

如果被设置为 True, 则可以使用 exception 对象。相反, 如果被设置为 false, 则不可以使用 exception 对象。

默认值是 false。

11. extends="package.class"

表明 JSP 编译时需要加入的 Java Class 的全名, 但是必须慎重地使用它, 它会限制 JSP 的编译能力。

例如:

```
<% page extends="com.myPackage.AservletImplementation" %>
```

#### 4.2.2 Include 指令

在 JSP 文件中用 Include 指令包含一个静态的文件, 同时解析这个文件中的 JSP 语句。基本语法:

```
<%@ include file="path" %>
```

包含文件的路径名一般来说是指相对路径, 不需要什么端口、协议和域名, 如下所示:

```
<%@ include file="header.inc"%>
```

如果路径以 "/" 开头, 那么路径主要是参照 JSP 应用的上下关系路径; 如果路径是以文件名或目录名开头, 那么这个路径就是正在使用的 JSP 文件的当前路径。

文件 index.jsp 的代码如例程 4-1 所示。

例程 4-1

```
<%@ include file="header.inc"%>
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="workM" scope="page" class="test.faq" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>当前位置: <a href="index.jsp">首页</a> </td>
  </tr>
</tbody>
<%@ include file="date.inc"%>
```

```

</tr>
<TR bgColor=#3399ff>
  <TD height=1 colspan="6"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
<tr><td height=10 colspan="6"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
</TBODY></TABLE>
<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">
      <%@ include file="table.inc"%>
    </td>
    <td width="10" height="100%"></td>
    <td width="1" height="100%" bgcolor="#3399ff"></td>
    <td width="10" height="100%"></td>
    <td width="589" height="331" valign="top" background="images/bg1.gif">
      <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
        <tr>
          <td width="100%" height="20" bgcolor="">&nbsp;
          <a href="mailto:blueriver_cn@sina.com">最新动态</a>
        </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
</td>
</tr>
</table>
<%@ include file="footer.inc"%>

```

其中，被 include 的文件 header.inc 的代码如例程 4-2 所示。

例程 4-2

```

<HTML><HEAD><TITLE></TITLE>
<META content="text/html; charset=gb2312" http-equiv=Content-Type>
<LINK href="css/site.css" rel=stylesheet>
<META content="MSHTML 5.00.2014.210" name=GENERATOR></HEAD>
<script language="javascript" src="../js/all.js">
</script>
<BODY bgColor=#ffffff leftMargin=0 topMargin=5 marginheight="5" marginwidth="5">
  <table align="center" border="0" width="760" height="18" bgcolor="#3399FF" cellspacing="0">
    <tr>
      <td width="100%">
        <p><a class="x" href="member/list_all_user.jsp">用户列表</a> |
        <a class="x" href="book_store/index.jsp">商品列表</a> |
        <a class="x" href="search/index.jsp">查询商品</a> |
        <a class="x" href="book_store/all_category.jsp">商品分类</a> |
        <a class="x" href="book_store/new_goods.jsp">新货上架</a> |
        <a class="x" href="book_store/good_price.jsp">特价市场</a> |

```

其中，被 include 的文件 date.inc 的代码如例程 4-3 所示。

### 例程 4-3

其中，被 include 的文件 table.inc 的代码如例程 4-4 所示。

### 例程 4-4

74

```

<td width="100%" bgcolor="#3399FF" height="18" align="center">
<font color="#ffffff">本站统计</font>
</td>
</tr>
<tr>
<td width="100%">
<table border="0" width="98%" align="center" cellspacing="0">
<tr>
<td align="right" height="20" width="60%">注册用户: </td>
<%
    int      t_member;t_member=0;
    String    strSQLmem="SELECT id FROM member";
    ResultSet RSmem = workM.executeQuery(strSQLmem);
    while(RSmem.next())
{
        t_member=t_member+1;
    }
%>
<td align="left" width="40%"><%=t_member%></td>
</tr>
.....
</table>

```

其中, 被 include 的文件 footer.inc 的代码如例程 4-5 所示。

例程 4-5

```

<hr align="center" color="#3399ff" width="760" >
<table align="center" border="0" width="760">
<tr>
<td width="100%" height="18" align="center">
<P class="foot">热爱 JSP 吧
<a href="mailto:blueriver_cn@sina.com">Blueriver</a>&nbsp;  <br><br></P>
</td>
</tr>
<tr><td>
</tr>
</table>
</body>
</html>

```

运行上述代码, 您将会看到如图 4-1 所示的效果图。

描述:

Include 指令将在 JSP 编译时插入一个包含文本或代码的文件, 当您使用 Include 指令时, 这个包含的过程就是静态的。静态的包含就是指这个被包含的文件将会被插入到 JSP 文件中去, 这个包含的文件可以是 JSP 文件、HTML 文件、文本文件、inc 文件等。如果包含的文件中包含可执行代码, 那么这个包含的文件中的代码将会被执行。

如果您仅仅是用 Include 来包含一个静态文件。那么这个包含的文件所执行的结果将



会插入到 JSP 文件中放`<% @ include %>`的地方。一旦包含文件被执行，那么主 JSP 文件的过程将会被恢复，继续执行下一行。

这个被包含文件可以是 html 文件、jsp 文件、文本文件或者只是一段 Java 代码，但是您必须注意，在这个包含文件中不能使用`<html>`、`</html>`、`<body>`、`</body>`标记，因为这将会影响在原 JSP 文件中同样的标记，这样做有时会导致错误。

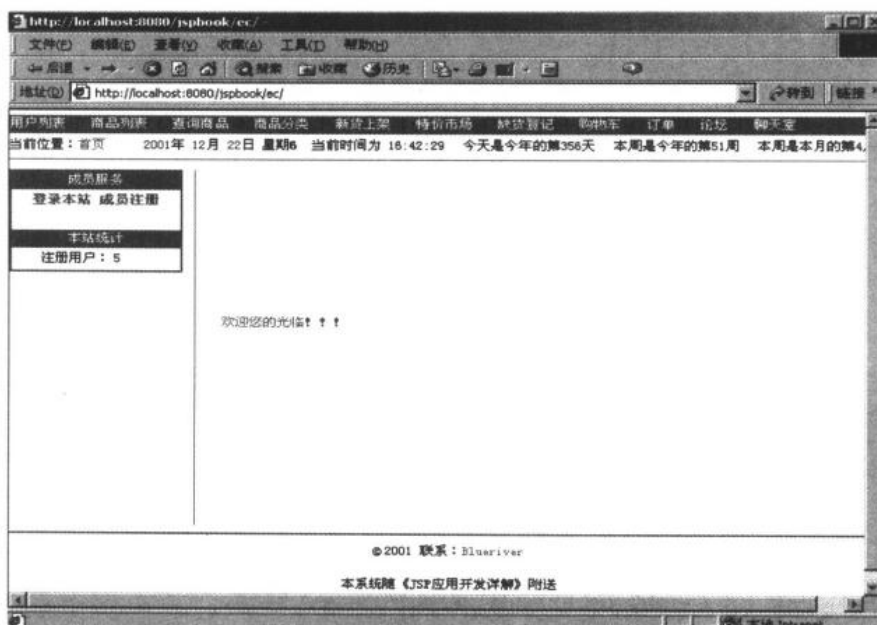


图 4-1 运行效果图

## 4.3 JSP 动作指令

上一节介绍了 JSP 的指令，这一节我们来重点介绍 JSP 的动作指令。

### 4.3.1 Include 指令

`<jsp:include>` 标签包含一个静态的或者动态的文件。

语法：

```
<jsp:include page="path" flush="True" />
or
<jsp:include page=" path " flush="True" >
    <jsp:param name="paramName" value="paramValue " />...
</jsp:include>
```

例如：

```
<jsp:include page="/templates/copyright.html " />
<jsp:include page=" templates/copyright.html ">
<jsp:include page="index.html" />
<jsp:include page="/index.html" />
```

```
<jsp:include page="copyright.html" />
<jsp:param name="companyName" value="Sun-Oracle-Java-One" />
</jsp:include>
```

其中:

1. `page="path"`

参数为一个相对路径, 或者是代表相对路径的表达式。如果路径以“/”开头, 那么路径主要是参照 JSP 应用的上下关系路径。如果路径是以文件名或目录名开头, 那么这个路径就是正在使用的 JSP 文件的当前路径。

2. `flush="True"`

这里您必须使用 `flush="True"`, 不能使用 `false` 值。默认值为 `false`。

3. `<jsp:param name="paramName" value="paramValue" />`

`<jsp:param>`子句能让您传递一个或多个参数给动态文件, 也可以在一个页面中使用多个`<jsp:param>`来传递多个参数给动态文件。

描述:

`<jsp:include>`标签允许您包含动态文件和静态文件, 但是这两种包含文件的结果是不同的。如果文件仅仅是静态文件, 那么这种包含仅仅是把包含文件的内容加到 jsp 文件中, 这个被包含文件不会被 Jsp 编译器执行。相反地, 如果这个被包含文件是动态的文件, 那么这个被包含文件将会被 Jsp 编译器执行。

`<jsp:include>`标签能够同时处理静态和动态两种文件, 因此不需要判断被包含文件是动态的还是静态的。

如果被包含文件是动态的, 那么还可以用`<jsp:param>`还传递参数名和参数值。

例如:

在 `tomcat/examples/jsp/include/`下的 `include.jsp`:

```
<html>
<body bgcolor="white">
<font color="red">
<%%@ page buffer="5" autoFlush="false" %>
<p>In place evaluation of another JSP which gives you the current time:
<%%@ include file="foo.jsp" %>
<p> <jsp:include page="/jsp/include/foo.html" flush="True"/> by including the output of another JSP:
<jsp:include page="foo.jsp" flush="True"/>
:-)
</html>
foo.jsp:
<body bgcolor="white">
<font color="red">
<%= System.currentTimeMillis() %>
```

文件 `foo.html` 代码如下:

To get the current time in ms

运行结果如图 4-2 所示。

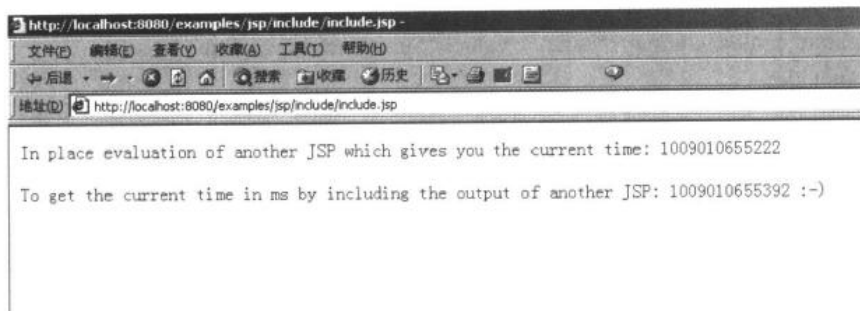


图 4-2 运行效果图

### 4.3.2 Forward 指令

<jsp:forward>标签重定向一个 HTML 文件、JSP 文件，或者是一个程序段。

JSP 语法：

```
<jsp:forward page="path" />
or
<jsp:forward page=" path " >
    <jsp:param name="paramName" value="paraValue " />....
</jsp:forward>
```

例如：

```
<% String whereTo="/templates/login.jsp"%>
<jsp:forward page="<%=whereTo%>" />
```

或者：

```
<jsp:forward page="/templates/login.jsp ">
    <jsp:param name="userID" value="blueriver" />
</jsp:forward>
```

其中：

1. page=" path "

" path " 值为一个表达式或者一个字符串，它用于说明将要定向的文件或 URL。这个文件可以是 JSP 文件，也可以是程序段，或者其他能够处理 request 对象的文件。

2. <jsp:param name=" paramName " value=" paraValue " />.....

name 指定参数名，value 指定参数值。参数被发送到一个动态文件，参数可以是一个或多个值，而这个文件却必须是动态文件。

如果要传递多个参数，则可以在一个 JSP 文件中使用多个<jsp:param>将多个参数发送到一个动态文件中。

描述：

<jsp:forward>标签从一个 JSP 文件向另一个文件传递一个包含用户请求的 request 对象。<jsp:forward>标签以下的代码，将不能执行。

可以使用<jsp:param>标签向目标文件传送参数和值，注意如果使用了<jsp:param>标签，那么目标文件必须是一个动态的文件。

如果使用了非缓冲输出,那么如果在使用<jsp:forward>之前,JSP 文件已经有了数据,文件执行就会出错。

例如:

文件 forward.jsp 代码如例程 4-6 所示。

例程 4-6

```
<html>
<%
    double freeMem = Runtime.getRuntime().freeMemory();
    double totlMem = Runtime.getRuntime().totalMemory();
    double percent = freeMem/totlMem;
    if (percent < 0.5)
%>
<jsp:forward page="/jsp/forward/one.jsp"/>
<% } else { %>
<jsp:forward page="two.html"/>
<% } %>
</html>
```

文件 one.jsp 代码如下:

```
<html>
<body bgcolor="white">
<font color="red">
VM Memory usage < 50%.
</html>
```

文件 two.html 代码如下:

```
<html>
<body bgcolor="white">
<font color="red">
VM Memory usage > 50%.
</html>
```

运行效果如图 4-3 所示。

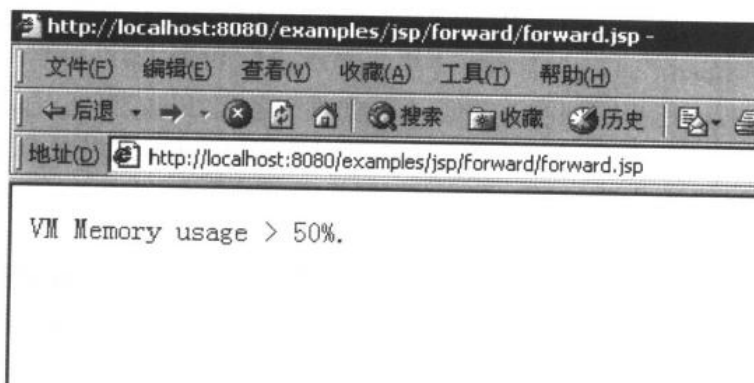


图 4-3 运行效果图

### 4.3.3 UseBean 指令

<jsp:useBean>标签用来在 JSP 页面中创建一个 Bean 实例并指定它的名字以及作用范围。

基本语法

<jsp:useBean id="name" scope="page | request | session | application" typeSpec />其中 typeSpec 有以下几种可能的情况:

```
class=" className" |
class=" className " type="typeName" |
beanName="beanName" type="typeName" |
type=" typeName " |
```

描述:

您必须使用 class 或 type, 而不能同时使用 class 和 BeanName。BeanName 表示 Bean 的名字, 其形式为"a.b.c"。<jsp:useBean>用于定位或示例一个 JavaBeans 组件。<jsp:useBean>首先会试图定位一个 Bean 实例, 如果这个 Bean 不存在, 那么<jsp:useBean>就会从一个 class 或模版中进行示例。

另外, <jsp:useBean>标签的主体部分通常包含有<jsp:setProperty>标签, 用于设置 Bean 的属性值。<jsp:useBean>的主体仅仅在<jsp:useBean>示例 Bean 时才会被执行, 如果这个 Bean 已经存在, <jsp:useBean>能够定位它, 那么主体中的内容将不再起作用。

属性:

#### 1. id="name"

在所定义的范围中确认 Bean 的变量, 以便能在后面的程序中使用此变量名来分辨不同的 Bean。这个变量名必须符合所使用的脚本语言的规定 (在 Java Programming Language 中, 这个规定在 Java Language 规范中已经写明), 而且变量名对大小写敏感。如果这个 Bean 已经在别的<jsp:useBean>中创建, 那么这个 id 的值必须与原来的那个 id 值一致。

#### 2. scope="page | request | session | application"

Bean 存在的范围以及 id 变量名的有效范围。以下是详细说明。

- page 能够在包含<jsp:useBean>标签的 JSP 文件以及此文件中的所有静态包含文件中使用 Bean, 直到页面执行完毕向客户端发回响应或转到另一个文件为止。
- request 在任何执行相同请求的 Jsp 文件中使用 Bean, 直到页面执行完毕向客户端发回响应或转到另一个文件为止。可以使用 Request 对象访问 Bean, 比如:  
request.getAttribute(name);

其中 name 是指 bean 实例化的名字。

- session 从创建 Bean 开始, 就可以在任何使用相同 session 的 Jsp 文件中使用 Bean。这个 Bean 存在于整个 Session 生存周期内, 任何在分享此 Session 的 Jsp 文件都能使用同一 Bean。

需要注意的是, 在创建 Bean 的 Jsp 文件中<% @ page %>指令中必须指定 session=True, 否则可能会导致致命的错误发生。

可以使用 session 对象访问 Bean, 比如:



```
session.getValue(name) ;
```

其中 name 是指 Bean 实例化的名字。

- application 同 session 一样,从创建 Bean 开始,就可以在任何使用相同 application 的 Jsp 文件中使用 Bean。这个 Bean 存在于整个 application 生存周期内,任何在分享此 application 的 Jsp 文件都能使用同一 Bean。

可以使用 application 对象访问 Bean, 比如:

```
application.getAttribute(name) ;
```

其中 name 是指 Bean 实例化的名字。

默认值是 page。

### 3. class=" className "

使用 new 关键字以及 class 构造器 (constructor) 从一个 class 中示例一个 Bean。这个 class 不能是抽象的,必须有一个公用的、没有参数的构造器 (constructor)。该 package 的名字对大小写敏感。

### 4. beanName=" beanName " type=" typeName "

使用 instantiate 方法从一个 class 中示例一个 Bean, 同时还可以指定 Bean 的类型。instantiate 方法在类 java.beans.Beans 中。

beanName 可以是 package 或 class, 也可以是表达式。

package 和 class 名字对大小写敏感。

### 5. type=" typeName "

type 可以是一个类本身 (class), 也可以是一个类的父类 (superclass of a class), 或者是一个类的接口 (interface implemented by the class)。如果您没有使用 class 或 BeanName 指定 type, Bean 将不会被示例。package 和 class 的名字对大小写敏感。

例如:

```
<jsp:useBean id="cart" scope="session" class="session.Carts" />
<jsp:setProperty name="cart" property="*" />
<jsp:useBean id="checking" scope="session" class="bank.Checking" >
<jsp:setProperty name="checking" property="balance" value="0.0" />
</jsp:useBean>
```

## 4.3.4 GetProperty 指令

这个属性获取 Bean 的属性的值并将之转化为一个字符串, 然后将其插入到输出的页面中。

语法:

```
<jsp:getProperty name=" name" property="propertyName" />
```

描述:

这个<jsp:getProperty>标签将获得 Bean 的属性值, 并将之转化为一个字符串, 然后显示在 JSP 页面中。

需要注意的是, 在使用<jsp:getProperty>之前, 必须用<jsp:useBean>创建它。

<jsp:getProperty>标签有一些限制:

- 不能使用<jsp:getProperty>来检索一个已经被索引了的属性。
- 能够和 JavaBeans 组件一起使用<jsp:getProperty>,但是不能与 Enterprise Java Bean 一起使用。

属性:

name=" name "

这是个必选属性。其值为 Bean 的名字,在这之前用 jsp:useBean 引入的名称。

property="propertyName"

这是个必选属性。其值为所指定的 Bean 的属性名。

例如:

```
<jsp:useBean id=" user " scope="session" class="company. user" />
<jsp:getProperty name="user" property="name" />
```

#### 4.3.5 SetProperty 指令

<jsp:setProperty>标签用来设置 Bean 中的属性值。

基本语法:

```
<jsp:setProperty name="beanName" prop_expr />
```

其中:

prop\_expr 有以下几种可能的情形:

```
property=" *" |
property=" propertyName" |
property=" propertyName" param=" parameterName"
property=" propertyName" value=" propertyValue"
```

描述:

在前面我们就知道了可以使用 jsp:setProperty 来为一个 Bean 的属性赋值。您可以使用两种方式来实现它。

一种是:在 jsp:useBean 后使用 jsp:setProperty。

```
<jsp:useBean id="myUser " ... />
...
<jsp:setProperty name="user" property=" user " ... />
```

在这种方式中, jsp:setProperty 将被执行,即使已经存在一个具有相同的 id 和 scope bean。

另一种是, jsp:setProperty 出现在 jsp:useBean 标签内:

```
<jsp:useBean id=" myUser " ... >
...
<jsp:setProperty name=" user " property=" user " ... />
</jsp:useBean>
```

在这种方式中, jsp:setProperty 只会在新的对象被实例化时才被执行。

<jsp:useBean>和<jsp:setProperty>是联系在一起的,同时它们使用的 Bean 实例的名字也应当匹配,即在<jsp:setProperty>中的 name 的值应当和<jsp:useBean>中 id 的值相同。

<jsp:setProperty>标签使用 Bean 给定的 setXXX()方法,在 Bean 中设置一个或多个属

性值。利用<jsp:setProperty>来设置属性值有多种方法。

- 通过用户输入的所有值来匹配 Bean 中的属性
- 通过用户输入的指定的值来匹配 Bean 中指定的属性
- 在运行时使用一个表达式来匹配 Bean 的属性

属性:

name 属性值是其属性将被设置的 Bean 实例的名称。jsp:usebean 必须出现在 jsp:setProperty 之前。

- property="\*"

在 Bean 中, 属性的名字、类型必须和 request 对象中的参数名称、类型相一致。该属性用来存储用户在 Jsp 中输入的所有值, 用于匹配 Bean 中的属性。

从客户传到服务器上的参数值一般都是字符类型, 这些字符串为了能够在 Bean 中匹配就必须转换成其他的类型, 表 4-1 中列出了 Bean 属性的类型以及它们的转换方法。

表 4-1 把字符串转化为其他类型的方法

Property 类型	方 法
boolean	java.lang.Boolean.valueOf(String).booleanValue()
Boolean	java.lang.Boolean.valueOf(String)
byte	java.lang.Byte.valueOf(String).byteValue()
Byte	java.lang.Byte.valueOf(String)
char	java.lang.Character.valueOf(String).charValue()
Character	java.lang.Character.valueOf(String)
double	java.lang.Double.valueOf(String).doubleValue()
Double	java.lang.Double.valueOf(String)
int	java.lang.Integer.valueOf(String).intValue()
Integer	java.lang.Integer.valueOf(String)
float	java.lang.Float.valueOf(String).floatValue()
Float	java.lang.Float.valueOf(String)
long	java.lang.Long.valueOf(String).longValue()
Long	java.lang.Long.valueOf(String)

如果 request 对象的参数值中有空值, 那么对应的 Bean 属性将不会设置任何值。同样, 如果 Bean 中有一个属性没有与之对应的 Request 参数值, 那么这个属性同样也不会设定。

如果使用了 property="\*", 那么 Bean 的属性没有必要按 Html 表单中的顺序排序。

- property="propertyName"

使用 request 中的一个参数值来指定 Bean 中的一个属性值。在这个语法中, property 指定 Bean 的属性名, 而且 bean 属性和 request 参数的名字应相同, 否则需要用另一种语法, 即指定 param。

如果 request 对象的参数值中有空值, 那么对应的 Bean 属性将不会设定任何值。

- property="propertyName" param="parameterName"

property 属性在前一个描述中已经讲述, 这里只重点讲述 param。

param 指定 request 中的参数名。其中需要注意的是, 当 bean 属性和 request 参数的名字不同时, 就必须得指定 property 和 param。如果它们同名, 那么您就只需要指明 property 即可。

如果 request 对象的参数值中有空值, 那么对应的 Bean 属性将不会设置任何值。

- property="propertyName" value="propertyValue"

value 是一个可选属性, 使用指定的值来设定 Bean 属性。这个值可以是字符串, 也可以是表达式。如果是字符串, 则字符串的值通过相应的对象或包的标准的 valueOf 方法将自动地转换为 Bean 属性的类型。

例如, boolean 或 Boolean 属性的值 “false” 将通过 java.lang.Boolean.valueOf(String) 方法转化, 而一个 int 或 Integer 属性的值 “79” 将通过 java.lang.Integer.valueOf() 转化。

如果是一个表达式, 那么它的类型就必须和它将要设定的属性值的类型一致。

如果参数值为空, 那么对应的属性值也不会被设定。

另外, 您不能在一个 <jsp:setProperty> 中同时使用 param 和 value。

#### 4.3.6 Plugin 指令

执行一个 Applet 或 Bean, 有可能的话还要下载一个 Java 插件用于执行它。

JSP 语法:

```
<jsp:plugin
    type="bean | applet"
    code="classFileName"
    codebase="classFileDirectoryName"
    [ name="instanceName" ]
    [ archive="URIToArchive, ..." ]
    [ align="bottom | top | middle | left | right" ]
    [ height="displayPixels" ]
    [ width="displayPixels" ]
    [ hspace="leftRightPixels" ]
    [ vspace="topBottomPixels" ]
    [ jreversion="JREVersionNumber | 1.1" ]
    [ nspluginurl="URLToPlugin" ]
    [ iepluginurl="URLToPlugin" ] >
    <jsp:params>
    [ <jsp:param name="parameterName" value="{parameterValue | <%= expression %>}" ]
/> ]+
</jsp:params> ]
[ <jsp:fallback> text message for user </jsp:fallback> ]
</jsp:plugin>
```

例如:

```
<jsp:plugin type=applet code="Molecule.class" codebase="/html">
    <jsp:params>
        <jsp:param name="molecule" value="molecules/benzene.mol" />
    </jsp:params>
</jsp:plugin>
```

```
</jsp:params>
<jsp:fallback>
    <p>Unable to load applet</p>
</jsp:fallback>
</jsp:plugin>
```

描述:

`<jsp:plugin>`元素用于在浏览器中播放或显示一个对象（典型的就昰 Applet 和 Bean），而这种显示需要有浏览器的 java 插件。

当 JSP 文件被编译，送往浏览器时，`<jsp:plugin>`元素将会根据浏览器的版本替换成 `<object>` 或者 `<embed>` 元素。

需要注意的是，`<object>`用于 HTML 4.0，`<embed>`用于 HTML 3.2。

一般来说，`<jsp:plugin>`元素会指定对象是 Applet 还是 Bean，同样也会指定 class 的名字，还有位置，另外还会指定将从哪里下载这个 Java 插件。具体如下：

属性：

1. `type="bean | applet"`

将被执行的插件对象的类型。

必须在 Bean 或 Applet 中指定一个，因为这个属性没有默认值。

2. `code="className"`

插件将执行的 Java 类文件的名称。在名称中您必须包含扩展名。且此文件必须在用“codebase”属性指明的目录下。

3. `codebase="classFileDirectoryName"`

包含插件将运行的 Java 类的目录或指向这个目录的路径。默认为此 JSP 文件的路径。

4. `name="instanceName"`

Bean 或 Applet 的实例的名称。使得被同一个 JSP 文件调用的 Bean 或 Applet 之间的通信成为可能。

5. `archive="URIToArchive, ....."`

以逗号分隔的路径名列表。是那些用以 codebase 指定的目录下的类装载机预装载的存档文件所在的路径名。

通常，这些存档文件通过网络被安全的加载，可以显著地提高 Applet 的性能。

6. `align="bottom | top | middle | left | right"`

图形、对象、Applet 的位置，有以下值：

bottom

top

middle

left

right

7. `height="displayPixels" width="displayPixels"`

Applet 或 Bean 将要显示的长宽的值，此值为数字，单位为像素。



8. `hspace="leftRightPixels" vspace="topBottomPixels"`

Applet 或 Bean 显示时在屏幕左右、上下所需留下的空间大小，单位为像素。

9. `jreversion="JREVersionNumber | 1.1"`

Applet 或 Bean 运行所需的 Java Runtime Environment (JRE) 的版本。默认值是 1.1。

10. `nspluginurl="URLToPlugin"`

Netscape Navigator 用户能够使用的 JRE 的下载地址，此值为一个标准的 URL，如 `http://www.sun.com`。

11. `iepluginurl="URLToPlugin"`

Internet Explorer 用户能够使用的 JRE 的下载地址，此值为一个标准的 URL，如 `http://www.sun.com`。

12. `<jsp:params>`

[ `<jsp:param name="parameterName" value="{parameterValue | <%= expression %>}"`  
`/> ]+ </jsp:params>`

需要向 Applet 或 Bean 传送的参数或参数值。

13. `<jsp:fallback> text message for user </jsp:fallback>`

用于 Java 插件不能启动时显示给用户的一段文字，如果插件能够启动而 Applet 或 Bean 不能，那么浏览器会有一个出错信息弹出。

文件 `plugin.jsp` 代码如例程 4-7 所示。

例程 4-7

```
<html>
<title> Plugin example </title>
<body bgcolor="white">
<h3> Current time is : </h3>
<jsp:plugin type="applet" code="Clock2.class" codebase="/examples/jsp/plugin/applet"
jreversion="1.2" width="160" height="150" >
  <jsp:fallback>
    Plugin tag OBJECT or EMBED not supported by browser.
  </jsp:fallback>
</jsp:plugin>
<p>
<h4>
<font color=red>
The above applet is loaded using the Java Plugin from a jsp page using the
plugin tag.
</font>
</h4>
</body>
</html>
```

`clock2.java` 源代码如例程 4-8 所示。

例程 4-8

```
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.text.*;

public class Clock2 extends Applet implements Runnable {
    Thread timer;
    int lastxs, lastys, lastxm, lastym, lastxh, lastyh;
    SimpleDateFormat formatter;
    String lastdate;
    Font clockFaceFont;
    Date currentDate;
    Color handColor;
    Color numberColor;
    public void init() {
        int x,y;
        lastxs = lastys = lastxm = lastym = lastxh = lastyh = 0;
        formatter = new SimpleDateFormat ("EEE MMM dd hh:mm:ss yyyy",
Locale.getDefault());
        currentDate = new Date();
        lastdate = formatter.format(currentDate);
        clockFaceFont = new Font("Serif", Font.PLAIN, 14);
        handColor = Color.blue;
        numberColor = Color.darkGray;
        try {
            setBackground(new Color(Integer.parseInt(getParameter("bgcolor"),16)));
        } catch (Exception E) { }
        try {
            handColor = new Color(Integer.parseInt(getParameter("fgcolor1"),16));
        } catch (Exception E) { }
        try {
            numberColor = new Color(Integer.parseInt(getParameter("fgcolor2"),16));
        } catch (Exception E) { }
        resize(300,300);           // Set clock window size
    }
    public void plotpoints(int x0, int y0, int x, int y, Graphics g) {
        g.drawLine(x0+x,y0+y,x0+x,y0+y);
        g.drawLine(x0+y,y0+x,x0+y,y0+x);
        g.drawLine(x0+y,y0-x,x0+y,y0-x);
        g.drawLine(x0+x,y0-y,x0+x,y0-y);
        g.drawLine(x0-x,y0-y,x0-x,y0-y);
        g.drawLine(x0-y,y0-x,x0-y,y0-x);
        g.drawLine(x0-y,y0+x,x0-y,y0+x);
        g.drawLine(x0-x,y0+y,x0-x,y0+y);
    }
    public void circle(int x0, int y0, int r, Graphics g) {
```

```
int x,y;
float d;
x=0;
y=r;
d=5/4-r;
plotpoints(x0,y0,x,y,g);
while (y>x){
    if (d<0) {
        d=d+2*x+3;
        x++;
    }
    else {
        d=d+2*(x-y)+5;
        x++;
        y--;
    }
    plotpoints(x0,y0,x,y,g);
}
}
// Paint is the main part of the program
public void paint(Graphics g) {
    int xh, yh, xm, ym, xs, ys, s = 0, m = 10, h = 10, xcenter, ycenter;
    String today;
    currentDate = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat("s",Locale.getDefault());
    try {
        s = Integer.parseInt(formatter.format(currentDate));
    } catch (NumberFormatException n) {
        s = 0;
    }
    formatter.applyPattern("m");
    try {
        m = Integer.parseInt(formatter.format(currentDate));
    } catch (NumberFormatException n) {
        m = 10;
    }
    formatter.applyPattern("h");
    try {
        h = Integer.parseInt(formatter.format(currentDate));
    } catch (NumberFormatException n) {
        h = 10;
    }
    formatter.applyPattern("EEE MMM dd HH:mm:ss yyyy");
    today = formatter.format(currentDate);
    xcenter=80;
    ycenter=55;
```

```

xs = (int)(Math.cos(s * 3.14f/30 - 3.14f/2) * 45 + xcenter);
ys = (int)(Math.sin(s * 3.14f/30 - 3.14f/2) * 45 + ycenter);
xm = (int)(Math.cos(m * 3.14f/30 - 3.14f/2) * 40 + xcenter);
ym = (int)(Math.sin(m * 3.14f/30 - 3.14f/2) * 40 + ycenter);
xh = (int)(Math.cos((h*30 + m/2) * 3.14f/180 - 3.14f/2) * 30 + xcenter);
yh = (int)(Math.sin((h*30 + m/2) * 3.14f/180 - 3.14f/2) * 30 + ycenter);
g.setFont(clockFaceFont);
g.setColor(handColor);
circle(xcenter,ycenter,50,g);
g.setColor(numberColor);
g.drawString("9",xcenter-45,ycenter+3);
g.drawString("3",xcenter+40,ycenter+3);
g.drawString("12",xcenter-5,ycenter-37);
g.drawString("6",xcenter-3,ycenter+45);
g.setColor(getBackground());
if (xs != lastxs || ys != lastys) {
    g.drawLine(xcenter, ycenter, lastxs, lastys);
    g.drawString(lastdate, 5, 125);
}
if (xm != lastxm || ym != lastym) {
    g.drawLine(xcenter, ycenter-1, lastxm, lastym);
    g.drawLine(xcenter-1, ycenter, lastxm, lastym); }
if (xh != lastxh || yh != lastyh) {
    g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
    g.drawLine(xcenter-1, ycenter, lastxh, lastyh); }
g.setColor(numberColor);
g.drawString("", 5, 125);
g.drawString(today, 5, 125);
g.drawLine(xcenter, ycenter, xs, ys);
g.setColor(handColor);
g.drawLine(xcenter, ycenter-1, xm, ym);
g.drawLine(xcenter-1, ycenter, xm, ym);
g.drawLine(xcenter, ycenter-1, xh, yh);
g.drawLine(xcenter-1, ycenter, xh, yh);
lastxs=xs; lastys=ys;
lastxm=xm; lastym=ym;
lastxh=xh; lastyh=yh;
lastdate = today;
currentDate=null;
}
public void start() {
    timer = new Thread(this);
    timer.start();
}
public void stop() {
    timer = null;

```

```
}  
public void run() {  
    Thread me = Thread.currentThread();  
    while (timer == me) {  
        try {  
            Thread.currentThread().sleep(100);  
        } catch (InterruptedException e) {  
        }  
        repaint();  
    }  
}  
public void update(Graphics g) {  
    paint(g);  
}  
public String getAppletInfo() {  
    return "Title: A Clock \nAuthor: Rachel Gollub, 1995 \nAn analog clock.";  
}  
public String[][] getParameterInfo() {  
    String[][] info = {  
        {  
            "bgcolor", "hexadecimal RGB number",  
            "The background color. Default is the color of your browser."  
        },  
        {  
            "fgcolor1", "hexadecimal RGB number",  
            "The color of the hands and dial. Default is blue."  
        },  
        {  
            "fgcolor2", "hexadecimal RGB number",  
            "The color of the seconds hand and numbers. Default is dark gray."  
        }  
    };  
    return info;  
}  
}
```

运行结果如图 4-4~图 4-5 所示。

plugin.jsp 经过服务器解释后产生的 HTML 文件源代码，如例程 4-9 所示。

例程 4-9

```
<html>  
<title> Plugin example </title>  
<body bgcolor="white">  
<h3> Current time is : </h3>  
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"  
    width="160" height="150"  
    codebase="http://java.sun.com/products/plugin/1.2.2/jinstall-1_2_2-win.cab#Version=1,2,2,0">
```



```
<PARAM name="java_code" value="Clock2.class">
<PARAM name="java_codebase" value="/examples/jsp/plugin/applet">
<PARAM name="type" value="application/x-java-applet;version=1.2">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.2" width="160" height="150"
pluginspage="http://java.sun.com/products/plugin/"
java_code="Clock2.class"
java_codebase="/examples/jsp/plugin/applet" >
<NOEMBED>
</COMMENT>
Plugin tag OBJECT or EMBED not supported by browser.
</NOEMBED></EMBED>
</OBJECT>

<p>
<h4>
<font color=red>
The above applet is loaded using the Java Plugin from a jsp page using the
plugin tag.
</font>
</h4>
</body>
</html>
```

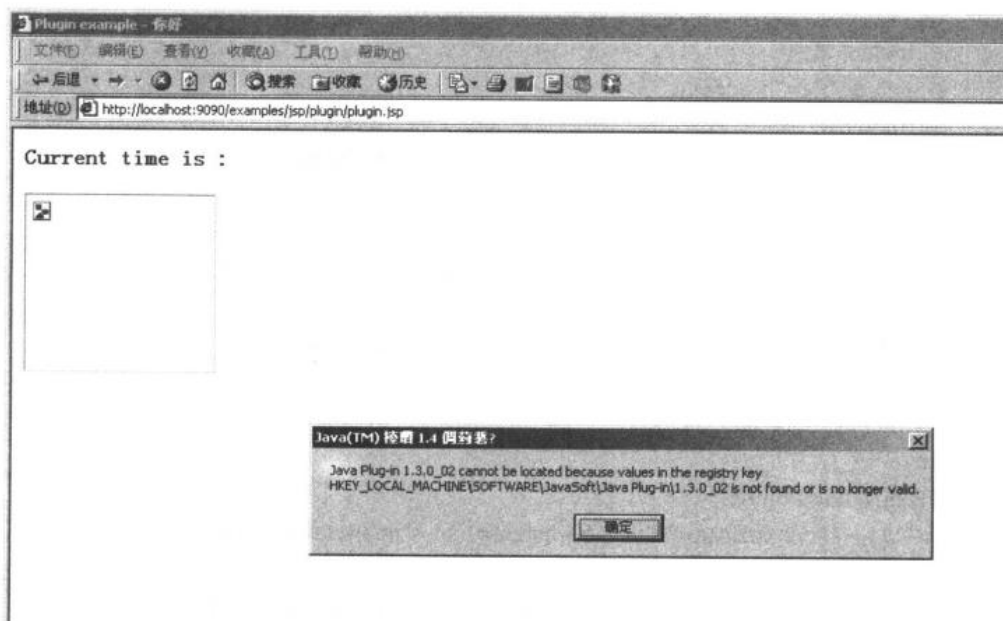


图 4-4 运行效果图

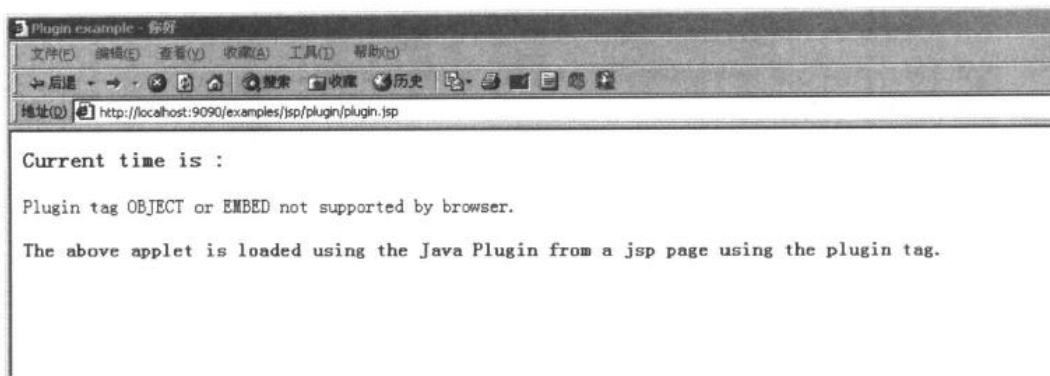


图 4-5 运行效果图

## 4.4 JSP 语法实例 NowTime

下面，将结合例子来讲述一下 JSP 的语法。

### 1. 文件 index.jsp (ec 根目录下)

文件 index.jsp 的源代码如例程 4-10 所示。

例程 4-10

```
//使用 include 指令
<%@ include file="header.inc"%>
// USEBEAN 指令
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="workM" scope="page" class="test.faq" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>当前位置: <a href="index.jsp">首页</a> </td>
//使用 include 指令
    <%@ include file="date.inc"%>
  </tr>
  <TR bgColor=#3399ff>
    <TD height=1 colspan="6"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
    <tr><td height=10 colspan="6"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
  </TBODY></TABLE>
  <table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
    <tr>
      <td width="150" height="355" valign="top">
//使用 include 指令
      <%@ include file="table.inc"%>
    </td>
  </tr>
</table>
```

```

</td>
<td width="10" height="100%"></td>
<td width="1" height="100%" bgcolor="#3399ff"></td>
<td width="10" height="100%"></td>
<td width="589" height="331" valign="top" background="images/bg1.gif">
<table border="0" width="100%" cellpadding="0" cellspacing="0" height="307">
<tr>
<td width="100%" height="20" bgcolor="">&nbsp;
<a href="mailto:blueriver_cn@sina.com">最新动态</a>
</td>
</tr>
</table>
</td>
</tr>
</table>
<%@ include file="footer.inc"%>

```

在 index.jsp 文件中 header.inc 文件:

```

<HTML><HEAD><TITLE></TITLE>
<META content="text/html; charset=gb2312" http-equiv=Content-Type>
<LINK href="css/site.css" rel=stylesheet>
<META content="MSHTML 5.00.2014.210" name=GENERATOR></HEAD>
<script language="javascript" src="../js/all.js">
</script>
<BODY bgColor=#ffffff leftMargin=0 topMargin=5 marginheight="5" marginwidth="5">
<table align="center" border="0" width="760" height="18" bgcolor="#3399FF" cellpadding="0">
<tr>
<td width="100%">
<p><a class="x" href="member/list_all_user.jsp">用户列表</a>|
<a class="x" href="book_store/index.jsp">商品列表</a>|
<a class="x" href="search/index.jsp">查询商品</a>|
<a class="x" href="book_store/all_category.jsp">商品分类</a>|
<a class="x" href="book_store/new_goods.jsp">新货上架</a>|
<a class="x" href="book_store/good_price.jsp">特价市场</a>|
<a class="x" href="book_store/short_goods.jsp">缺货登记</a>|
<a class="x" href="book_store/shopcart.jsp" target="_blank">购物车</a>|
<a class="x" href="book_store/order.jsp" target="_blank">订单</a>|
<a class="x" href="forum/index.jsp" target="_blank">论坛</a>|
<a class="x" href="chat/index.jsp" target="_blank">聊天室</a></td>
</tr>
</table>

```

## 2. 文件 footer.inc

在 index.jsp 文件中的 footer.inc 文件的源代码如例程 4-11 所示。

例程 4-11

```
<hr align="center" color="#3399ff" width="760" >
```

```

<table align="center" border="0" width="760">
  <tr>
    <td width="100%" height="18" align="center">
      <P class="foot">热爱 JSP 吧
      <a href="mailto:blueriver_cn@sina.com">Blueriver</a>&nbsp;  <br><br></P>
    </td>
  </tr>
</table>
</body>
</html>

```

### 3. 文件 table.inc

在 index.jsp 文件中的 table.inc 文件的源代码如例程 4-12 所示。

例程 4-12

```

<table border="1" width="100%" bordercolorlight="#3399FF"
bordercolordark="#3399FF" cellspacing="0" cellpadding="0">
  <tr>
    <td width="100%" valign="top" style="border: 1 solid #3399FF">
      <table border="0" width="100%" cellspacing="0">
        <tr>
          <td width="100%" bgcolor="#3399FF" height="18" align="center">
            <font color="#ffffff">成员服务</font></td>
          </tr>
          <tr>
            <td width="100%" height="23" align="center">
              <a href="member/login.jsp"><b>登录本站</b></a>
              <a href="member/reg.jsp"><b>成员注册</b></a></td>
            </tr>
            <td height="18">
              <tr>
                <td width="100%" bgcolor="#3399FF" height="18" align="center">
                  <font color="#ffffff">本站统计</font></td>
                </tr>
                <tr>
                  <td width="100%">
                    <table border="0" width="98%" align="center" cellspacing="0">
                      <tr>
                        <td align="right" height="20" width="60%">注册用户: </td>
                        <%
                          int t_member;
                          t_member=0;
                          String strSQLmem="SELECT id FROM member";
                          ResultSet RSmem = workM.executeQuery(strSQLmem);
                          while(RSmem.next()){
                            t_member=t_member+1;
                          }
                        %>

```

.....以下代码省略

文件 `date.inc` 的源代码如例程 4-13 所示。

[illegible]

class 文件 JspCalendar.java 的源代码如例程 4-14 所示。

```
package dates;

import java.text.DateFormat;
import java.util.*;

public class JspCalendar {

    Calendar calendar = null;

    public JspCalendar() {
        calendar = Calendar.getInstance();
        Date trialTime = new Date();
        calendar.setTime(trialTime);
    }

    public int getYear() {
        return calendar.get(Calendar.YEAR);
    }

    public String getMonth() {
        int m = getMonthInt();
        String[] months = new String [] { "1", "2", "3",
                                           "4", "5", "6",
                                           "7", "8", "9",
```



```
        "10", "11", "12" };

    if (m > 12)
        return "Unknown to Man";
    return months[m - 1];
}

public String getDay() {
    int x = getDayOfWeek();
    String[] days = new String[] { "1", "2", "3",
                                    "4", "5", "6", "7" };

    if (x > 7)
        return "Unknown to Man";
    return days[x - 1];
}

public int getMonthInt() {
    return 1 + calendar.get(Calendar.MONTH);
}

public String getDate() {
    return getMonthInt() + "/" + getDayOfMonth() + "/" + getYear();
}

public String getTime() {
    return getHour() + ":" + getMinute() + ":" + getSecond();
}

public int getDayOfMonth() {
    return calendar.get(Calendar.DAY_OF_MONTH);
}

public int getDayOfYear() {
    return calendar.get(Calendar.DAY_OF_YEAR);
}

public int getWeekOfYear() {
    return calendar.get(Calendar.WEEK_OF_YEAR);
}

public int getWeekOfMonth() {
    return calendar.get(Calendar.WEEK_OF_MONTH);
}

public int getDayOfWeek() {
    return calendar.get(Calendar.DAY_OF_WEEK) - 1;
}

public int getHour() {
    return calendar.get(Calendar.HOUR_OF_DAY);
}

public int getMinute() {
    return calendar.get(Calendar.MINUTE);
}
```

```
public int getSecond() {
    return calendar.get(Calendar.SECOND);
}

public static void main(String args[]) {
    JspCalendar db = new JspCalendar();
    p("date: " + db.getDayOfMonth());
    p("year: " + db.getYear());
    p("month: " + db.getMonth());
    p("time: " + db.getTime());
    p("date: " + db.getDate());
    p("Day: " + db.getDay());
    p("DayOfYear: " + db.getDayOfYear());
    p("WeekOfYear: " + db.getWeekOfYear());
    p("era: " + db.getEra());
    p("ampm: " + db.getAMPM());
    p("DST: " + db.getDSTOffset());
    p("ZONE Offset: " + db.getZoneOffset());
    p("TIMEZONE: " + db.getUSTimeZone());
}

private static void p(String x) {
    System.out.println(x);
}

public int getEra() {
    return calendar.get(Calendar.ERA);
}

public String getUSTimeZone() {
    String[] zones = new String[] { "Hawaii", "Alaskan", "Pacific",
                                     "Mountain", "Central", "Eastern" };
    return zones[10 + getZoneOffset()];
}

public int getZoneOffset() {
    return calendar.get(Calendar.ZONE_OFFSET)/(60*60*1000);
}

public int getDSTOffset() {
    return calendar.get(Calendar.DST_OFFSET)/(60*60*1000);
}

public int getAMPM() {
    return calendar.get(Calendar.AM_PM);
}

}
```

运行效果如图 4-6 所示。

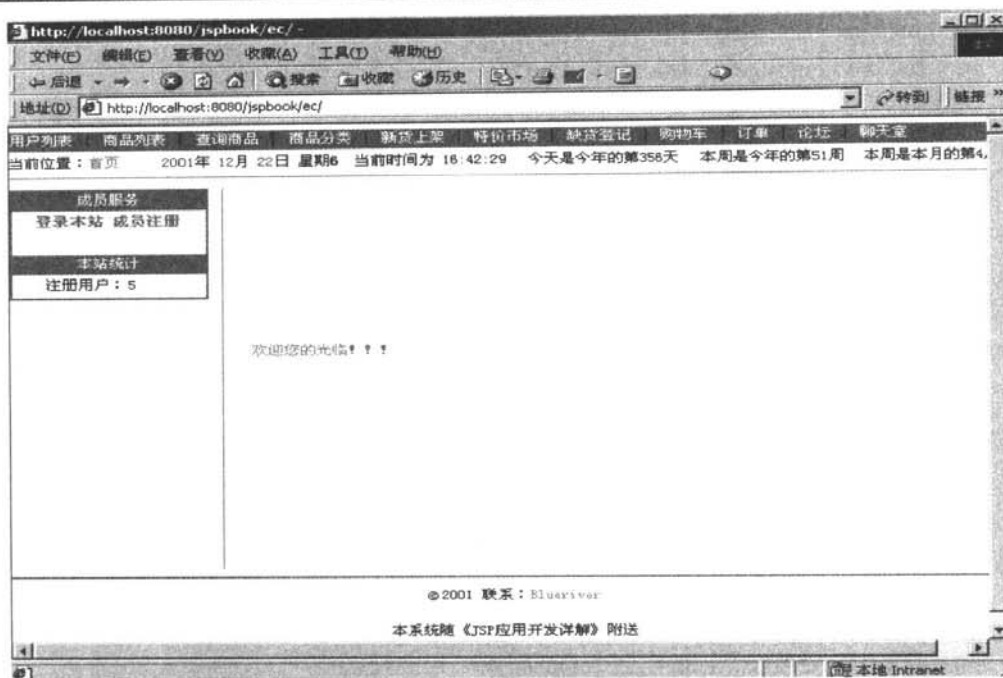


图 4-6 运行效果图

## 第 5 章 JSP 的内建对象及实例分析

在本章中，讲述了 JSP 的内建对象及实例分析。首先讲述了 JSP 内建对象的基本概念，接着又讲述了 JSP 的内建对象 request、response、out、Session、pageContext、application、config、page，最后一节则讲述了一个实例。

通过对本章的学习，将会掌握 JSP 的内建对象在开发中的应用。

### 5.1 JSP 的内建对象概述

JSP 的内建对象有以下几种：request、response、out、Session、pageContext、application、config 和 page。

- request

与 request 相联系的是 HttpServletRequest 类。通过 getParameter 方法可以得到 request 的参数，通过 GET、POST、HEAD 等方法可以得到 request 的类型，通过 cookies, Referer, 等可以得到引入的 HTTP 头。

严格来说，request 是类 javax.servlet.ServletException 的一个子类，而不是 HttpServletRequest 类。

- response

与 response 相联系的是 HttpServletResponse 类。因为输出流是放入缓冲的，所以可以设置 HTTP 状态码和 response 头。

- out

使用 PrintWriter 类来向客户端发送输出。然而，为了使 response 对象有效，可使用一个 PrintWrite 类的使用缓冲的版本 JspWriter。

使用 Session 的属性 page directive，您可以自己定义缓冲的大小，甚至可以在使用了 buffer 属性后关闭缓冲。

- Session

与 Session 相联系的是 HttpSession 类。Session 是自动创建的，即使没有一个引入的 Session，这种变量仍可绑定。

特别注意的是：

如果用 page 指令关闭 Session，而再试图使用 Session 时将导致错误，因此不要轻易地去关闭 Session。

- pageContext

这是 JSP 中的一个新的类 PageContext，PageContext 对象的主要功能是管理对属于 JSP 中特殊可见部分中已命名对象的访问。

PageContext 对象的创建与初始化，通常对 JSP 程序员是透明的，JSP 程序员可以从 JSP

中获取到用来代表 `PageContext` 对象的句柄，因此也就可以使用 `PageContext` 对象的各种 API。

- `application`

与 `application` 相联系的是 `ServletContext` 类，通过使用 `getServletConfig().getContext()` 方法得到。

`Application` 是一个很重要的对象。一旦创建了 `application` 对象，`application` 对象将会一直保持下去，直到服务器关闭为止。

- `config`

是一个 `ServletConfig` 类的对象。

JSP 配置处理程序的句柄，只有在 JSP 页面范围之内才是合法的。

- `page`

在 JAVA 中不是很有用，它仅仅是用来保存在脚本语言不是 JAVA 时的时间。

JSP 实现类对象的一个句柄，只有在 JSP 页面的范围之内才是合法的。

当使用 Java 作为脚本编程语言时，对象名 `this` 也可以用来引用这个对象。

### 5.1.1 JSP 的内建对象一：request

`request` 对象的主要方法有：

- `getAttribute(String name)`

返回 `name` 指定的属性值，如果指定的属性值不存在，则会返回一个 `null` 值。

- `getAttributeNames()`

返回 `request` 对象的所有属性的名字，其结果是一个类举（`Enumeration`）类的实例。

- `getCookies()`

返回客户端的 `Cookie` 对象，结果是一个 `Cookie` 数组。

- `getHeader(String name)`

获得 http 协议定义的文件头信息。

- `getHeaders(String name)`

返回指定名字的 `request Header` 的所有值，其结果也是一个类举（`Enumeration`）类的实例。

- `getHeaderNames()`

返回所有 `request Header` 的名字，其结果也是一个类举（`Enumeration`）类的实例。

- `getMethod()`

获得客户端向服务器端传送数据的方法，如 `get`、`post` 等。

- `getParameter(String name)`

获得客户端传送给服务器端的参数值，该参数是由 `name` 指定的。

- `getParameterNames()`

获得客户端传送给服务器端的所有参数的名字，其结果也是一个类举（`Enumeration`）类的实例。

- `getParameterValues(String name)`

获得指定参数的所有值，参数由 name 指定。

- `getProtocol()`

获取客户端向服务器端传送数据所依据的协议名称。

- `getQueryString()`

获得查询字符串，该字符串是由客户端以 `get` 方法向服务器端传送的。

- `getRequestURI()`

获取发出请求字符串的客户端地址。

- `getRemoteAddr()`

获取客户端的 IP 地址。

- `getRemoteHost()`

获取客户端的名字。

- `getServerName()`

获取服务器的名字。

- `getServletPath()`

获取客户端所请求的脚本文件的文件的路径。

- `getServerPort()`

获取服务器的端口号。

- `setAttribute(String name, java.lang.Object objt)`

设置名字为 name 的 request 参数的值，该值是由 `java.lang.Object` 类型的 objt 指定。  
snoop.JSP (examples/JSP/snp/snoop.JSP) 文件的源代码如例程 5-1 所示。

例程 5-1

```
<html>
<body bgcolor="white">
<h1> Request Information </h1>
<font size="4">
JSP Request Method: <%= request.getMethod() %>
<br>
Request URI: <%= request.getRequestURI() %>
<br>
Request Protocol: <%= request.getProtocol() %>
<br>
Servlet path: <%= request.getServletPath() %>
<br>
Path info: <%= request.getPathInfo() %>
<br>
Path translated: <%= request.getPathTranslated() %>
<br>
Query string: <%= request.getQueryString() %>
<br>
Content length: <%= request.getContentLength() %>
<br>
Content type: <%= request.getContentType() %>
```



```
<br>
Server name: <%= request.getServerName() %>
<br>
Server port: <%= request.getServerPort() %>
<br>
Remote user: <%= request.getRemoteUser() %>
<br>
Remote address: <%= request.getRemoteAddr() %>
<br>
Remote host: <%= request.getRemoteHost() %>
<br>
Authorization scheme: <%= request.getAuthType() %>
<hr>
The browser you are using is <%= request.getHeader("User-Agent") %>
<hr>
</font>
</body>
</html>
```

运行结果如图 5-1 所示。

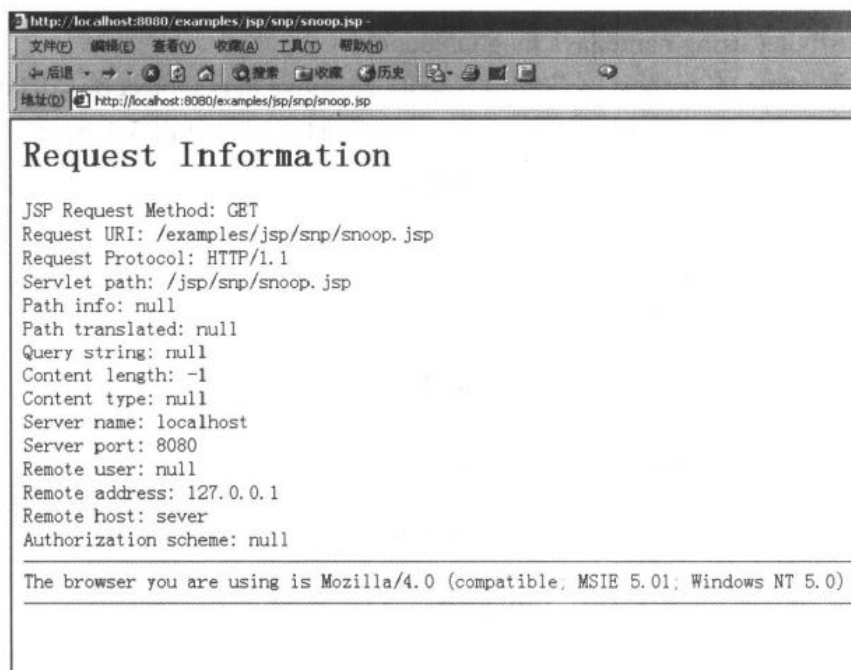


图 5-1 运行结果图

提交 Form 内的信息的源代码如例程 5-2 所示。

例程 5-2

```
<%@ include file="head.inc"%>
<%if(Session.getAttribute("username")==null){
    response.sendRedirect("../member/login.JSP?url="+request.getRequestURI());
```

```

}
%>

<JSP:useBean id='clock' scope='page' class='dates.JspCalendar' type='dates.JspCalendar' />

<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left"
height=25><%if(Session.getAttribute("username")!=null){ out.println(Session.getAttribute("username")); }
%> 当前位置: <a href="../index.JSP">首页</a> -&gt; <a href="index.JSP">张贴</a> -&gt; 张贴
</td>
    <%@ include file="../member/date.inc"%>
  </tr>
  <TR bgColor=#3399ff>
    <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
    <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
  </TBODY></TABLE>

<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">

    </td>
    <td width="10" height="100%"></td>
    <td width="1" height="100%" bgcolor="#3399ff"></td>
    <td width="10" height="100%"></td>
    <td width="589" height="331" valign="top" background="images/bg1.gif">
    <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
      <tr>
        <td width="100%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff">张贴
</font>
      </td>
    </tr>
  </table>
  <tr>
    <td>
      <form name="form1" method="post" action="post_ok.JSP">
        <table width="100%" border="0" cellspacing="0" cellpadding="0">
          <tr><td colspan="3" height="20"></td></tr>
          <tr>
            <td align="right" width="120" valign="top">说明: </td>
            <td width="20">&nbsp;</td>
            <td width="450">
              <ul type="square">

```

都会被还原! </li>

<li>全部内容必须填</li>

```
</ul>
</td>
</tr>
<tr>
<td align="right">书名: </td>
<td width="20">*</td>
<td width="500">
<input type="text" name="name" size="64" value="">
</td>
</tr>
<tr>
<td align="right">作者: </td>
<td width="20">*</td>
<td width="500">
<input type="text" name="author" size="64" value="">
</td>
</tr>
<tr>
<td align="right">语言: </td>
<td width="20">*</td>
<td width="500">
<input type="radio" name="language" value="1" checked>中文
<input type="radio" name="language" value="2">英文
</td>
</tr>
<tr>
<td align="right">类别: </td>
<td width="20">*</td>
<td width="500">
<input type="radio" name="category" value="1" checked>计算机类
<input type="radio" name="category" value="2">英语类
<input type="radio" name="category" value="3">其他类
</td>
</tr>
<tr>
<td align="right" valign="top">内容简介: </td>
<td width="20" valign="top">*</td>
<td>
<textarea name="content" cols="64" rows="5"></textarea>
</td>
</tr>
```

```
<tr>
  <td align="right">推荐程度: </td>
  <td width="20">*</td>
  <td width="500">
    <input type="radio" name="commend" value="1">好
    <input type="radio" name="commend" value="2">较好
    <input type="radio" name="commend" value="3" checked>一般
    <input type="radio" name="commend" value="4">较差
    <input type="radio" name="commend" value="5">差
  </td>
</tr>
<tr>
  <td align="right">出版社名称: </td>
  <td width="20">*</td>
  <td width="500">
    <input type="text" name="publish_name" size="64" value="">
  </td>
</tr>
<tr>
  <td align="right">出版社地址: </td>
  <td width="20">*</td>
  <td width="500">
    <input type="text" name="publish_address" size="64" value="">
  </td>
</tr>
<tr>
  <td align="right">是否配有光盘: </td>
  <td width="20">*</td>
  <td width="500">
    <input type="radio" name="cdrom" value="1" checked>有
    <input type="radio" name="cdrom" value="0">没有
  </td>
</tr>
<tr>
  <td align="right">价格: </td>
  <td width="20">*</td>
  <td width="500">
    <input type="text" name="price1" size="4" value="">元
    <input type="text" name="price2" size="4" value="">角
  </td>
</tr>
<tr>
  <td align="right">是否是特价书: </td>
  <td width="20">*</td>
```

```

        <td width="500">
            <input type="radio" name="good_price" value="1" checked>是
            <input type="radio" name="good_price" value="0">否
        </td>
    </tr>
    <tr>
        <td align="right">库存量: </td>
        <td width="20">*</td>
        <td width="500">
            <input type="text" name="book_number" size="64" value="">
        </td>
    </tr>
    <tr>
        <td align="right">&nbsp;</td>
        <td width="20">&nbsp;</td>
        <td align="right">
            <input type="button" name="post" value="张贴" onclick="sub()">
        </td>
    </tr>
</table>
</form>
<%//}%>

    </td>
</tr>
</table>

<script language="javascript">
function sub()
{
    if(document.form1.name.value=="")
    {
        window.alert("请输入书名! ");
        document.form1.name.focus();
        return False;
    }
    if(document.form1.author.value=="")
    {
        window.alert("请输入作者姓名! ");
        document.form1.author.focus();
        return False;
    }
    if(document.form1.content.value=="")
    {
        window.alert("请输入内容简介! ");
        document.form1.content.focus();
        return False;
    }
}

```

```
    }
    if(document.form1.publish_name.value=="")
    {
        window.alert("请输入出版社名称! ");
        document.form1.publish_name.focus();
        return False;
    }
    if(document.form1.publish_address.value=="")
    {
        window.alert("请输入出版社地址! ");
        document.form1.publish_address.focus();
        return False;
    }

    if(document.form1.price1.value=="")
    {
        window.alert("请输入价格! ");
        document.form1.price1.focus();
        return False;
    }

    if(isNaN(parseInt(document.form1.price1.value))||document.form1.price1.value<0)
    {
        window.alert("请输入正确的价格! ");
        document.form1.price1.focus();
        return False;
    }
    if(document.form1.price2.value=="")
    {
        window.alert("请输入价格! ");
        document.form1.price2.focus();
        return False;
    }

    if(isNaN(parseInt(document.form1.price2.value))||document.form1.price2.value>9||document.form1.p
rice2.value<0)
    {
        window.alert("请输入正确的价格! ");
        document.form1.price2.focus();
        return False;
    }
    if(document.form1.book_number.value=="")
    {
        window.alert("请输入库存量! ");
        document.form1.book_number.focus();
        return False;
    }
```



```

    }

    if(isNaN(parseInt(document.form1.book_number.value))||document.form1.book_number.value<0)
    {
        window.alert("请输入正确的库存量! ");
        document.form1.book_number.focus();
        return False;
    }
    document.form1.submit();
}
</script>

</td>
</tr>

</table>
<%% include file="../member/footer.inc"%>
post_ok.JSP:
<%% page language="java" import="java.sql.*" %>
<JSP:useBean id="workM" scope="page" class="test.faq" />

<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {

    }
    return "null";
}
%>
<%!
public static String returnToBr(String sStr)
if (sStr == null || sStr.equals(""))
return sStr;
}

StringBuffer sTmp = new StringBuffer();

```

```
int i = 0;

while (i <= sStr.length()-1)
if (sStr.charAt(i) == '\n' || sStr.charAt(i) == '\r')
sTmp = sTmp.append("<br>");
} else if (sStr.charAt(i) == ' ')
sTmp = sTmp.append("&nbsp;");
}else
sTmp = sTmp.append(sStr.substring(i,i+1));
}

i++;
}
String S1;
S1=sTmp.toString();
return S1;
}

%>
<%!
public static String returnToHTML(String sStr)
if (sStr == null || sStr.equals(""))
return sStr;
}

StringBuffer sTmp1 = new StringBuffer();
int i = 0;

while (i <= sStr.length()-1)

if (sStr.charAt(i) == '<')
sTmp1 = sTmp1.append("&lt;");
} else if (sStr.charAt(i) == '>')
sTmp1 = sTmp1.append("&gt;");
}else
{
sTmp1 = sTmp1.append(sStr.substring(i,i+1));
}

i++;
}
String S2;
S2=sTmp1.toString();
return S2;
}
```

```

%>
<%!
String name,author,content,category,publish_name,publish_address;
String language,commend,price,book_number;
String cdrom,good_price;
%>
<%
name=returnToBr(returnToHTML(request.getParameter("name")));
author=returnToBr(returnToHTML(request.getParameter("author")));
content=returnToBr(returnToHTML(request.getParameter("content")));
category=returnToBr(returnToHTML(request.getParameter("category")));
publish_name=returnToBr(returnToHTML(request.getParameter("publish_name")));
publish_address=returnToBr(returnToHTML(request.getParameter("publish_address")));
language=returnToBr(returnToHTML(request.getParameter("language")));
commend=returnToBr(returnToHTML(request.getParameter("commend")));
price=returnToBr(returnToHTML(request.getParameter("price1")+ "." +request.getParameter("price2")
));
book_number=returnToBr(returnToHTML(request.getParameter("book_number")));
cdrom=returnToBr(returnToHTML(request.getParameter("cdrom")));
good_price=returnToBr(returnToHTML(request.getParameter("good_price")));

//转换成中文
name=getStr(name);
author=getStr(author);
content=getStr(content);
category=getStr(category);
publish_name=getStr(publish_name);
publish_address=getStr(publish_address);
%>

<%@ include file="head.inc"%>

<JSP:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />

<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left"
height=25><%if(Session.getAttribute("username")!=null){ out.println(Session.get
Attribute("username"));}%> 当前位置: <a href=" ../index.JSP">首页</a> -&gt; <a href="index.JSP">教
程文章</a> -&gt; 张贴教程文章  </td>
    <%@ include file=" ../member/date.inc"%>
  </tr>
  <TR bgColor=#3399ff>
    <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
    <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"

```

```

width=16></td></tr>
</TBODY></TABLE>

<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">
      <%//*****%>
      <%@ include file="./install/simple_left.inc"%>
      <%//*****%>

    </td>
    <td width="10" height="100%"></td>
    <td width="1" height="100%" bgcolor="#3399ff"></td>
    <td width="10" height="100%"></td>
    <td width="589" height="331" valign="top" background="images/bg1.gif">
      <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
        <tr>
          <td width="100%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff">书库
</font>

          </td>
        </tr>

        <tr>
          <td>
            <%=name+author+content+category+publish_name+publish_address+language+commend+price+book_number+cdrom+good_price%>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<%
  String sqlinsert="insert into book(name,author, language, content, category,
commend,publish_name, publish_address, cdrom,price, good_price, book_number) Values( '"+name+"',
 '"+author+"', '"+language+"', '"+content+"', '"+category+"', '"+commend+"', '"+publish_name +'",
 '"+publish_address+"', '"+cdrom+"', '"+price+"', '"+good_price+"', '"+book_number+"')";
  out.println(sqlinsert);
  workM.executeQuery(sqlinsert);
%>
<%@ include file="./member/footer.inc"%>

```

### 5.1.2 JSP 的内建对象二：response

response 对象用于向客户端发送数据。

response 对象的主要方法有：

- `addCookie(Cookie cook)`

添加一个 Cookie 对象，用来保存客户端的用户信息。

- `addHeader(String name,String value)`

添加 HTTP 文件头信息，该 Header 将传到客户端去，如果已经同名的 Header 存在，则将会覆盖已有的 Header。

- `containsHeader(String name)`

判断指定名字的 HTTP 文件头是否已经存在，然后返回真假布尔值。

- `sendError(int)`

向客户端发送错误的信息。

例如：404 是指网页不存在或者请求的页面无效。

- `setHeader(String name, String value)`

设置指定名字的 HTTP 文件头的值，如果该值已经存在，则新值会覆盖原有的旧值。

例如：利用 `response` 对象重定向

```
<%
.....
String url;
url=request.getParameter("url");
if(url==null)
{
    response.sendRedirect("http://localhost:9090/test/ec");
}
else{
    response.sendRedirect("http://localhost:9090"+url);
}
.....
%>
```

例如：定时刷新（refresh.JSP）

```
<!--本页用来说明 response 对象-->
<%@ page import="java.util.Date"%>
<html>
<head>
<title>定时刷新页面</title>
</head>
<body>
<b><font color="Black">本页用来说明 response 对象</font></b>
<br>
<b>当前时间为: </b>
<%
response.setHeader("refresh","10");
%>
<%
out.println(new Date());
%>
</body>
```

</html>

运行效果如图 5-2 所示。

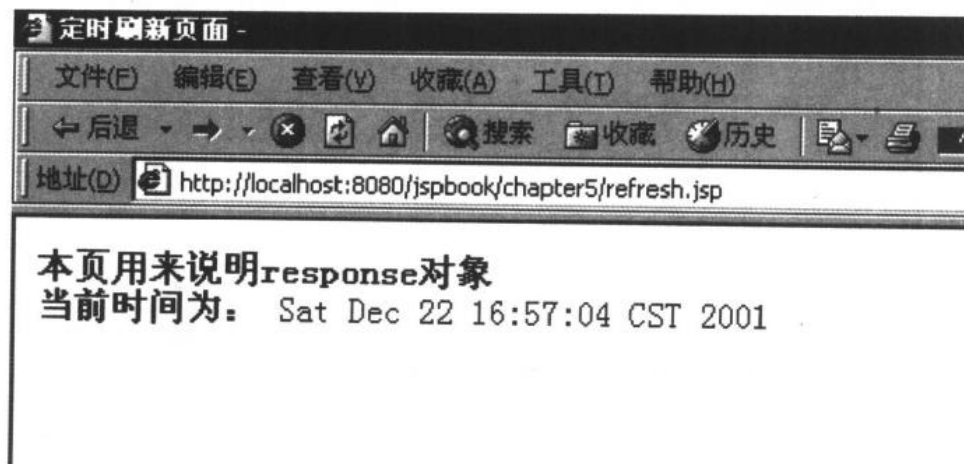


图 5-2 运行效果图

### 5.1.3 JSP 的内建对象三: out

Out 对象用来向客户端输出数据。

主要的方法有:

out.print(boolean)、out.println(boolean): 输出 boolean 类型的数据。

out.print(char)、out.println(char): 输出 char 类型的数据。

out.print(char[])、out.println(char[]): 输出 char[] 类型的数据。

out.print(double)、out.println(double): 输出 double 类型的数据。

out.print(float)、out.println(float): 输出 float 类型的数据。

out.print(int)、out.println(int): 输出 int 类型的数据。

out.print(long)、out.println(long): 输出 long 类型的数据。

out.print(Object)、out.println(Object): 输出 Object 类型的数据。

out.print(String)、out.println(String): 输出 String 类型的数据。

out.newLine(): 输出一个换行字符。

out.flush(): 输出缓冲区里的数据。

out.close(): 关闭输出流。

out.clearBuffer(): 清除缓冲区里的数据, 并把数据输出到客户端。

out.clear(): 清除缓冲区里的数据, 但不会把数据输出到客户端。

out.getBufferSize(): 获得缓冲区的大小。

out.getRemaining(): 获得缓冲区中没有被占用的空间的大小。

out.isAutoFlush(): 返回布尔值。如果 auto flush 为真, 则返回 True; 反之, 返回 False。

例如:

在发布信息时, 如果输入的信息包含中文字符, 如图 5-3 所示。



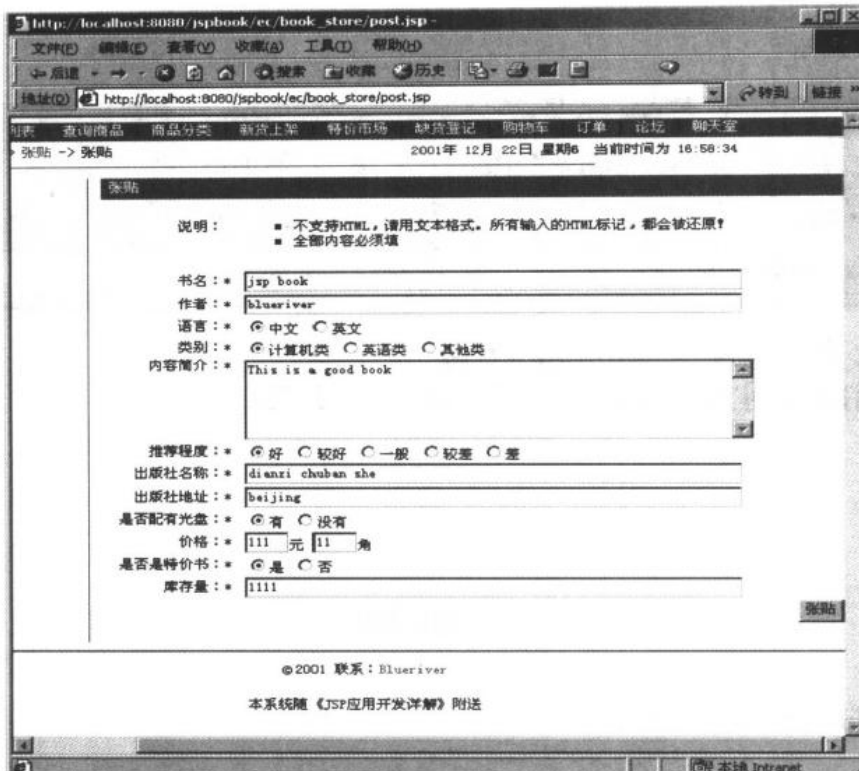


图 5-3 发布信息界面

提交后出现如图 5-4 所示的页面。值得注意的是，如果不解决中文内码转换的话，则可能提交的中文字符全变为“??”。

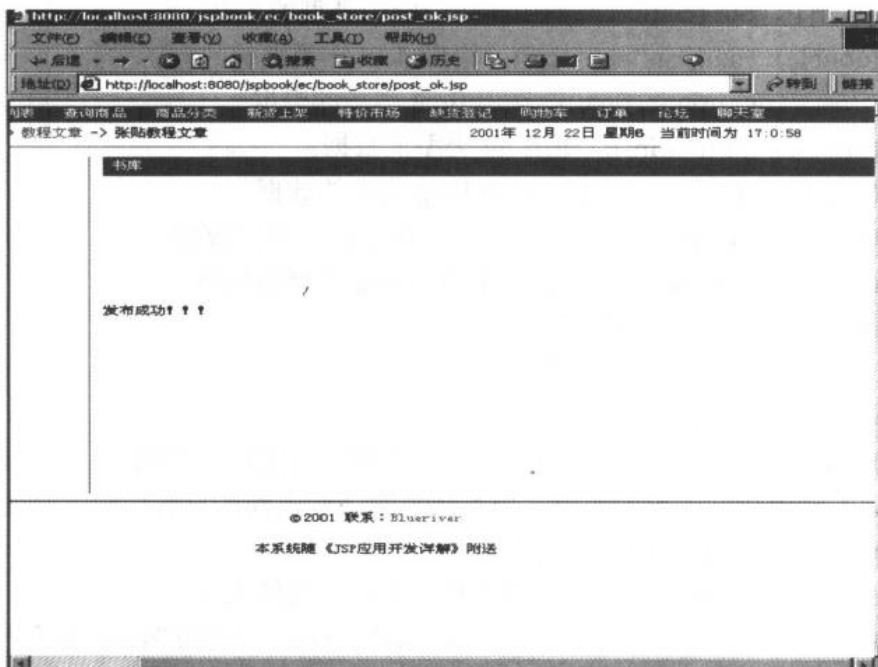


图 5-4 发布成功界面

提交后的页面 post\_ok.JSP 文件的源代码如例程 5-3 所示。

例程 5-3

```
<%@ page language="java" import="java.sql.*" %>
<JSP:useBean id="workM" scope="page" class="test.faq" />

<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
        }
    }
    return "null";
}
%>
<%!
public static String returnToBr(String sStr)
if (sStr == null || sStr.equals(""))
return sStr;
}

StringBuffer sTmp = new StringBuffer();
int i = 0;

while (i <= sStr.length()-1)
if (sStr.charAt(i) == '\n' || sStr.charAt(i) == '\r')
sTmp = sTmp.append("<br>");
} else if (sStr.charAt(i) == ' ')
sTmp = sTmp.append("&nbsp;");
} else
sTmp = sTmp.append(sStr.substring(i,i+1));
}

i++;
}
String S1;
S1=sTmp.toString();
return S1;
```

```
}

%>
<%!
public static String returnToHTML(String sStr)
if (sStr == null || sStr.equals(""))
return sStr;
}

StringBuffer sTmp1 = new StringBuffer();
int i = 0;

while (i <= sStr.length()-1)

if (sStr.charAt(i) == '<')
sTmp1 = sTmp1.append("&lt;");
} else if (sStr.charAt(i) == '>')
sTmp1 = sTmp1.append("&gt;");
}else
{
sTmp1 = sTmp1.append(sStr.substring(i,i+1));
}

i++;
}
String S2;
S2=sTmp1.toString();
return S2;
}

%>
<%!
String name,author,content,category,publish_name,publish_address;
String language,commend,price,book_number;
String cdrom,good_price;
%>
<%
name=returnToBr(returnToHTML(request.getParameter("name")));
author=returnToBr(returnToHTML(request.getParameter("author")));
content=returnToBr(returnToHTML(request.getParameter("content")));
category=returnToBr(returnToHTML(request.getParameter("category")));
publish_name=returnToBr(returnToHTML(request.getParameter("publish_name")));
publish_address=returnToBr(returnToHTML(request.getParameter("publish_address")));
language=returnToBr(returnToHTML(request.getParameter("language")));
commend=returnToBr(returnToHTML(request.getParameter("commend")));
price=returnToBr(returnToHTML(request.getParameter("price1")+ "." +request.getParameter("price2")))
```

```

));
book_number=returnToBr(returnToHTML(request.getParameter("book_number")));
cdrom=returnToBr(returnToHTML(request.getParameter("cdrom")));
good_price=returnToBr(returnToHTML(request.getParameter("good_price")));

//转换成中文
name=getStr(name);
author=getStr(author);
content=getStr(content);
category=getStr(category);
publish_name=getStr(publish_name);
publish_address=getStr(publish_address);
%>

<%@ include file="head.inc"%>

<JSP:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />

<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left"
height=25><%if(Session.getAttribute("username")!=null){ out.println(Session.getAttribute("username")); }
%> 当前位置: <a href=" ../index.JSP">首页</a> -&gt; <a href="index.JSP">教程文章</a> -&gt; 张贴
教程文章 </td>
    <%@ include file=" ../member/date.inc"%>
    </tr>
    <TR bgColor=#3399ff>
      <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
      <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
    </TBODY></TABLE>

<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">
      <%//*****%>
      <%@ include file=" ../install/simple_left.inc"%>
      <%//*****%>

    </td>
    <td width="10" height="100%"></td>
    <td width="1" height="100%" bgcolor="#3399ff"></td>
    <td width="10" height="100%"></td>
    <td width="589" height="331" valign="top" background="images/bg1.gif">
      <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">

```

```

        <tr>
            <td width="100%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff"> 书库
        </font>
        </td>
        </tr>

        <tr>
            <td>
                <%=name+author+content+category+publish_name+publish_address+language+commend+price+book_number+cdrom+good_price%>
            </td>
        </tr>
    </table>

    </td>
</tr>

</table>
<%
    String sqlinsert="insert into  book(name,author,language,content,category,commend,publish_name,
publish_address,cdrom,price,good_price,book_number)
Values('"+name+"','"+author+"','"+language+"','"+content+"','"+category+"','"+commend+"','"+publish_name+"','"+publish_address+"','"+cdrom+"','"+price+"','"+good_price+"','"+book_number+"')";
    out.println(sqlinsert);
    workM.executeQuery(sqlinsert);
%>
<%@ include file="../member/footer.inc"%>

```

解决中文内码转换的方法大致有以下几种:

如果是 request 传递的中文字符串,则必须用 getStr(String name)函数或其他内码转换函数来转换内码,否则会出现不能正常显示的情况。

如果是直接赋值的中文字符串,则不需要用 getStr(String name)函数或其他内码转换函数来转换内码,相反使用函数转换,会出现乱码,如上述例子。解决的方法就是直接在源码中加入<% @page contentType="text/html;charset=gb2312"%>即可。

#### 5.1.4 JSP 的内建对象四: Session

Session 对象是用来分别保存每一个用户信息的对象,以便于跟踪用户的操作状态。Session 的信息保存在服务器端,Session 的 ID 保存在客户机的 Cookie 中。事实上,在许多服务器上,如果浏览器支持 Cookies 的话就直接使用 Cookies。但是如果不支持或废除了 Cookies 的话就自动转化为 URL-rewriting,Session 自动为每个流程提供了方便地存储信息的方法。

不同的用户对应的 Session 对象一般是不相同的。例如当用户登录站点时，系统就会为他建立一个与其他不相同的 Session 对象，以便于区别其他用户。这个 Session 对象记录该用户的个人信息，而当该用户退出网站时，该 Session 对象就会随之消失。

Session 对象的主要方法有：

1. `getAttribute (String name)`

获取与指定名字 `name` 相联系的信息。

在 JSP1.0 中，这个方法为 `Session.getValue(String name)`。

2. `getAttributeNames ()`

返回 Session 对象中存储的每一个属性对象，其结果为一个枚举类 (Enumeration) 的实例。

3. `getCreationTime()`

返回 Session 被创建的时间。最小单位为千分之一秒。

为得到一个对打印输出很有用的值，可将此值传给 `Date constructor` 或者 `GregorianCalendar` 的方法 `setTimeInMillis`。

4. `getId()`

此方法返回惟一的标识，这些标识为每个 Session 而产生。当只有一个单一的值与一个 Session 联合时，或当日志信息与先前的 Sessions 有关时，它被当做键名用。

5. `GetLastAccessedTime()`

返回当前 Session 对象最后被客户发送的时间。最小单位为千分之一秒。

6. `GetMaxInactiveInterval()`

返回总时间 (秒)，负值表示 Session 永远不会超时。该时间为该 Session 对象的生存时间。

7. `removeAttribute(String name)`

删除与指定名字 `name` 的相联系的信息。

8. `setAttribute(String name, java.lang.Object value)`

设置指定名字 `name` 的属性值 `value`，并将之存储在 Session 对象中。

在 JSP1.0 中，这个方法为 `Session.putValue(String name, java.lang.Object value)`。在应用中使用最多的是 `getAttribute` 和 `setAttribute` 方法。

下面是用户登录的例程 5-4。

例程 5-4

```
<%
...
    if(rowcount!=0)
    {
        errorMessage="成功登录";
        Session.setAttribute("username",logname);
        loginOK=True;

        if(loginOK){
            String(olName+"<br><a          href=user_info.JSP?id="+Session.getAttribute("username")+"
```



```
target=_blank">"+Session.getAttribute("username")+"</a>");

String url;
url=request.getParameter("url");
out.println(url);
if(url==null){
    response.sendRedirect("http://localhost:9090/test/ec");
}
else{
    response.sendRedirect("http://localhost:9090"+url);
}
}
}else{
    errorMessage="您的用户名或者密码不正确";
    Session.setAttribute("username","");
}

...
%>
```

### 5.1.5 JSP 的内建对象五：pageContext

**pageContext** 对象的主要功能是管理对属于 JSP 中特殊可见部分中已经命名对象的访问。**PageContext** 对象的创建与初始化通常都对 JSP 程序员是透明的，JSP 程序员可以从 JSP 中获取到用来代表 **pageContextA** 对象的句柄，因此也就可以使用 **pageContext** 对象的各种 API。

在请求被服务的过程中，将会调用 **pageContext.initialize()** 方法，从而使得 **pageContext** 对象可以被 JSP 实现类所处理。对于一个特定的 JSP，许多传递给 **initialize()** 方法的信息都被直接从 **PAGE** 指令中获取。

**pageContext** 对象可以用来管理对能够从 JSP 进行引用的对象的访问。每个对象的范围对于考虑对象访问以及定义对象应用的合法性范围都是重要的。

**pageContext** 对象的 **getXXX()**、**setXXX()** 和 **findXXX()** 方法都可以用来根据不同的对象范围实现对这些对象的管理。**PageContext.getAttribute()** 方法可以用来获取默认的页面范围之中的一个已经命名对象的句柄，或者用来在另一个页面范围中获取一个已经命名对象的句柄。

**getAttribute scope()** 方法用来检索一个特定的已经命名的对象的范围，并且还可以通过调用 **getAttributeNamesInScope()** 方法，检索对某个特定范围的每个属性 **String** 字符串名称的枚举。

**findAttribute()** 方法则可以用来按照页面、请求、会话以及应用程序范围的顺序实现对某个已经命名属性的搜索。

**setAttribute()** 方法可以用来设置默认页面范围或特定对象范围之中的已命名对象。

**removeAttribute()** 方法可以用来删除默认页面范围或特定对象范围之中的已命名对象。

### 5.1.6 JSP 的内建对象六: application

application 对象用来在多个程序中保存信息, 每个用户的 application 对象都是相同的, 每一个用户都共用同一个 application 对象。这跟前面讲述的 Session 对象是不同的。

服务器启动后, 一旦创建了 application 对象, 那么这个 application 对象将会永远保持下去, 直到服务器关闭为止。

application 对象的主要方法:

1. `getAttribute(String name)`

返回由 name 指定的名字的 application 对象的属性的值。

2. `getAttributeNames()`

返回所有的 application 对象的属性的名字, 其结果是一个枚举 (Enumeration) 的实例。

3. `getInitParameter(String name)`

返回由 name 指定的名字的 application 对象的某个属性的初始值。

4. `getServletInfo()`

返回 servlet 编译器的当前版本的信息。

5. `setAttribute(String name, Object object)`

设置由 name 指定的名字的 application 对象的属性的值 object。

现以一个简单统计在线人数的例程来说明 Application 的应用 (这里不考虑离开的情况)。

init.jsp 文件的源代码如例程 5-5 所示。

例程 5-5

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<BODY BGCOLOR="#FFFFFF">
<%
//统计在线人数
application.setAttribute("counter",new Integer(0));
out.println(application.getAttribute("counter"));
//记录在线人名
application.setAttribute("onlineName",new String(""));
out.println(application.getAttribute("onlineName"));
%>
</BODY>
</HTML>
count.JSP(用来统计总人数并显示到页面):
<HTML>
<HEAD>
<TITLE> New Document </TITLE>

</head>
```

```
<BODY BGCOLOR="#FFFFFF" onUnload="javascript:cancel();">
<%
Integer i=(Integer)application.getAttribute("counter");
String name=(String)application.getAttribute("onlineName");
//StringBuffer total=new StringBuffer(name);
%>

<font color="blue">
<%out.println((String)Session.getAttribute("username"));%>您好:
</font>
<br><font color="blue">
当前在线人数: <%=i%>
</font>
<br>
<font color="red">
在线人总名册: <%out.println(name);%>
</font>

</BODY>
</HTML>
```

如果服务器支持 global.jsa 文件, 那么在 global.jsa 文件中完成 application、Session 对象的构造和注销工作。

在服务器启动或者 JSP 程序创建了 application、Session 对象的时候, 服务器会自动执行 global.jsa 文件, 并且完成 application、Session 对象的初始化工作。

当服务器关闭或者 JSP 程序中删除了 application、Session 对象的时候, 服务器会自动执行 global.jsa 文件, 并且完成 application、Session 对象的注销化工作。

笔者曾写过一個在程序中注销的例子, 读者可以做以参考。

本例的目的是: 当用户关闭该页时, 会自动注销; 读者可以根据这个例程做扩展, 以增强功能。

online.jsp 文件的源代码如例程 5-6 所示。

例程 5-6

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>

</head>
<BODY BGCOLOR="#FFFFFF" onUnload="javascript:cancel();">
<%
Integer i=(Integer)application.getAttribute("counter");
String name=(String)application.getAttribute("onlineName");
%>
```

```
<br><font color="blue">
当前在线人数: <%=i%>
</font>
<br>
<font color="red">
在线人总名册: <%out.println(name);%>
<font color="blue">
<%//out.println((String)Session.getAttribute("username"));%>
</font>
</font>
<script language="javascript">
<!--
function cancel(){
<%
if(Session.getAttribute("username")!= "" && Session.getAttribute("username")!= null){
//cancelname 为注销的人名
String cancelname;
cancelname=(String)Session.getAttribute("username");
//out.println(cancelname);

//j 为 cancelname 在总名册中的初始位置
//k 为 cancelname 的总长度
//t 为 cancelname 在总名册中的结束位置
int j,k,t;
j=name.indexOf(cancelname);
k=cancelname.length();
t=j+k;
//out.println(j);
//out.println(t);
///总名词中去除 cancelname
StringBuffer total=new StringBuffer(name);
total=total.delete(j-4,t);
//总人数减 1
i=new Integer(i.intValue()-1);
application.setAttribute("counter",i);
name=(String)total.toString();
application.setAttribute("onlineName",name);
//out.println("window.alert(""+name+"");

Session.setAttribute("username","");
}
%>
}
-->
</script>
</BODY>
```

&lt;/HTML&gt;

### 5.1.7 JSP 的内建对象七：config

config 对象是一个 `ServletConfig` 类的对象。config 对象主要用来配置处理 JSP 程序的句柄，而且只有在 JSP 页面范围之内才是合法的。

### 5.1.8 JSP 的内建对象八：page

page 对象在 Java 中不是很有用，它仅仅是用来保存在脚本的语言不是 Java 时的时间。JSP 实现类对象的一个句柄，只有在 JSP 页面的范围之内才是合法的。  
当使用 Java 作为脚本编程语言时，对象名 `this` 也可以用来引用这个对象。

## 5.2 JSP 的内建对象实例（简单的购物车）

shopcart.JSP 文件（在 ec/根目录下）的源代码如例程 5-7 所示。

例程 5-7

```
<%@ page language="java" import="java.sql.*" %>
<JSP:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.JSP"%>
<%
if(Session.getAttribute("username")==null||Session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.JSP?url="+request.getRequestURI());
}

%>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
    }
}
```

```

        return "null";
    }
    %>

<%
String book_id=request.getParameter("book_id");
//是否已经选择该货物
String sql="select ID from orders where book_id="+book_id+" and user_name="+Session.getAttribute("username")+"" and status=0";
ResultSet RS=workM.executeQuery(sql);
//out.println(sql);
int rowcount=0;
try
{
    while(RS.next())
    {
        rowcount++;
    }
}
catch(Exception e)
{
}

//把货物放入购物车

if(rowcount==0)
{
    String sqlBook1="select book.* from book where id="+book_id;
    ResultSet RSBook1=workM.executeQuery(sqlBook1);
    while(RSBook1.next())
    {
        String sqlCart="insert into orders(user_name,book_id,book_number,status,goods_price)
Values('"+Session.getAttribute("username")+",""+book_id+",'1','0','"+RSBook1.getDouble("price")+"'");
        workM.executeQuery(sqlCart);
    }
}
else
{
    String sqlBook2="select book.* from book where id="+book_id;
    ResultSet RSBook2=workM.executeQuery(sqlBook2);
    while(RSBook2.next())
    {
        String sqlAdd="update orders set book_number=book_number+1,goods_price=
"+RSBook2.getDouble("price")+"" where book_id="+book_id+" and user_name="+Session.getAttribute("username")+"" and status=0";

```



```

        workM.executeQuery(sqlAdd);
    }
}

int count;
count=0;

String sqlBook3="select * from price where user_name="+Session.getAttribute
("username");
ResultSet RSBook3=workM.executeQuery(sqlBook3);
while(RSBook3.next())
{
    count=count+1;
}
if(count==0)
{
    String sqlBook4="select * from orders where user_name="+Session.getAttribute
("username");

    ResultSet RSBook4=workM.executeQuery(sqlBook4);
    while(RSBook4.next())
    {
        String sqlPrice="insert into price(user_name,goods_price,total_price)
values('"+Session.getAttribute("username")+","+(double)(RSBook4.getDouble("goods_price")*RS-
Book4.getInt("book_number"))+"','goods_price+5')";
        workM.executeQuery(sqlPrice);
    }
}
else
{
    String sqlBook5="select * from orders where user_name="+Session.getAttribute
("username");

    ResultSet RSBook5=workM.executeQuery(sqlBook5);
    while(RSBook5.next())
    {
        String sqlPrice="update price set goods_price="+((double)(RSBook5.getDouble
("goods_price")*RSBook5.getInt("book_number"))+"total_price=goods_price+5";
        workM.executeQuery(sqlPrice);
    }
}

response.sendRedirect("cart.JSP");
%>
<html>
<head>
<title>[详细资料]</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">

```

```

</head>

<body bgcolor="#FFFFFF">
<div align="center">
  <table width="100%" border="0" bgcolor="#CCCCFF">
    <tr>
      <td>
        <div align="center"><b><font color="#FF0000">购 物 车</font></b></div>
      </td>
    </tr>
  </table>

  <p>您的购物车中包含以下货物:</p>
  <table width="100%" border="0" align="center">
    <tr bgcolor="#FFCCCC">
      <td width="15%">
        <div align="center"><font color="#0000FF">购买数量</font></div>
      </td>
      <td width="56%">
        <div align="center"><font color="#0000FF">书名</font></div>
      </td>
      <td width="13%">
        <div align="center"><font color="#0000FF">单价</font></div>
      </td>
      <td width="16%">
        <div align="center"><font color="#0000FF">总价格</font></div>
      </td>
    </tr>
    <%
      String sqlList="select * from orders where user_name='"+Session.getAttribute("username")+"'
and status=0";
      ResultSet RSList=workM.executeQuery(sqlList);
      try
      {
        while(RSList.next())
        {
          int b_num;
          b_num=RSList.getInt("book_number");
        }
      }
    %>
    <tr bgcolor="#CCFFCC">
      <td width="15%">
        <div align="center"><font color="#0000FF">
          <input type="text" name="book_number" size="4" value=<%=b_num%>>
        </font></div>
      </td>
    <%

```

```

String sqlBook="select book.* from book where id="+RSList.getInt("book_id");
ResultSet RSBook=workM.executeQuery(sqlBook);
while(RSBook.next())
{

    %>
    <td width="56%">
        <div align="center"><font color="#0000FF"><%=getStr(RSBook.getString("name"))
%></font> </div>
    </td>
    <td width="13%">
        <%
            double price;
            price=RSBook.getDouble("price");
        . %>
        <div align="center"><font color="#0000FF"><%=price%>元</font></div>
    </td>

    <td width="16%">
        <div align="center"><font color="#0000FF">&yen;<%=(double)price*b_num%> 元
</font></div>
    </td>
    <%}%>
</tr>
<tr bgcolor="#CCCCFF">
    <td colspan="3">
        <div align="center"><font color="#0000FF">货物价格</font></div>
    </td>
    <td width="16%">
        <div align="center"><font color="#0000FF">&yen;</font></div>
    </td>
</tr>
<%
    }
}
catch(Exception e)
{
}
%>
<tr>
    <td height="18" colspan="3" bgcolor="#CCCC99"><font color="#0000FF">&nbsp;</font>
        </td>
    <td align="center"><font color="#0000FF">运输费用</font></div>
    </td>
    <td height="18" width="16%" bgcolor="#CCCC99">
        <div align="center"><font color="#0000FF">&yen;</font></div>

```

```

        </td>
    </tr>
    <tr bgcolor="#66FF66">
        <td colspan="3">
            <div align="center"><b><font color="#FF0000">总费用</font></b></div>
        </td>
        <td width="16%">
            <div align="center"><b><font color="#FF0000">&yen;</font></b></div>
        </td>
    </tr>
    <tr align="center">
        <td colspan="4">
            <form name="form1" method="post" >
            <input type="button" value="继续购物" onclick="javascript:self.close();">
            <input type="submit" name="Submit" value="重新计算价格">
            <input type="submit" name="Submit2" value="填写订单">
            </form>
        </td>
    </tr>
</table>
<p>&nbsp;</p>
</div>

</body>
</html>

```

cart.jsp 文件的源代码如例程 5-8 所示。

例程 5-8

```

<%@ page language="java" import="java.sql.*" %>
<JSP:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.JSP"%>
<%
if(Session.getAttribute("username")==null||Session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.JSP?url="+request.getRequestURI());
}

%>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");

```

```

        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {

    }
    return "null";
}
%>

<html>
<head>
<title>[详细资料]</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body bgcolor="#FFFFFF">
<div align="center">
    <table width="100%" border="0" bgcolor="#CCCCFF">
        <tr>
            <td>
                <div align="center"><b><font color="#FF0000">购 物 车</font></b> </div>
            </td>
        </tr>
    </table>
    <form name="form1" method="post" action="order1.JSP">
    <p>您的购物车中包含以下货物: </p>
    <table width="100%" border="0" align="center">
        <tr bgcolor="#FFCCCC">
            <td width="15%">
                <div align="center"><font color="#0000FF">购买数量</font></div>
            </td>
            <td width="56%">
                <div align="center"><font color="#0000FF">书名</font></div>
            </td>
            <td width="13%">
                <div align="center"><font color="#0000FF">单价</font></div>
            </td>
            <td width="16%">
                <div align="center"><font color="#0000FF">总价格</font></div>
            </td>
        </tr>
    </table>
    <%

```

```

//价格
double g_price,total_price;
g_price=0;
total_price=0;
String sqlList="select * from orders where user_name='"+Session.getAttribute("username")+"'
and status=0";
ResultSet RSList=workM.executeQuery(sqlList);
try
{
    while(RSList.next())
    {
        int b_num;
        b_num=RSList.getInt("book_number");
        %>
        <tr bgcolor="#CCFFCC">
        <td width="15%">
            <div align="center"> <font color="#0000FF"> <%=b_num%>
            <input type="hidden" name="book_number" size="4" value=<%=b_num%>>
            </font></div>
        </td>
        <%
            String sqlBook="select book.* from book where id="+RSList.getInt("book_id");
            ResultSet RSBook=workM.executeQuery(sqlBook);
            while(RSBook.next())
            {

                %>
                <td width="56%">
                    <div align="center"><font color="#0000FF"><%=getStr(RSBook.getString("name"))
%></font></div>
                </td>
                <td width="13%">
                    <%
                        double price;
                        price=RSBook.getDouble("price");
                    %>
                    <div align="center"><font color="#0000FF">&yen;<%=price%></font></div>
                </td>

                <td width="16%">
                    <div align="center"><font color="#0000FF">&yen;<%= (float)price*b_num%>
</font></div>
                </td>
                <%
                    g_price=g_price+(double)price*b_num;
                %>
            }
        %>
    }
}

```



1

```
<p>&nbsp;</p>
</div>

</body>
</html>
recart.JSP:
<%@ page language="java" import="java.sql.*" %>
<JSP:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.JSP"%>
<%
if(Session.getAttribute("username")==null||Session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.JSP?url="+request.getRequestURI());
}

%>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {

    }
    return "null";
}
%>

<html>
<head>
<title>[详细资料]</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body bgcolor="#FFFFFF">
<div align="center">
    <table width="100%" border="0" bgcolor="#CCCCFF">
```

```

<tr>
  <td>
    <div align="center"><b><font color="#FF0000">购 物 车</font></b></div>
  </td>
</tr>
</table>
<form name="form1" method="post" action="">
<p>您的购物车中包含以下货物: </p>
<table width="100%" border="0" align="center">
  <tr bgcolor="#FFCCCC">
    <td width="15%">
      <div align="center"><font color="#0000FF">购买数量</font></div>
    </td>
    <td width="56%">
      <div align="center"><font color="#0000FF">书名</font></div>
    </td>
    <td width="13%">
      <div align="center"><font color="#0000FF">单价</font></div>
    </td>
    <td width="16%">
      <div align="center"><font color="#0000FF">总价格</font></div>
    </td>
  </tr>
  <%
    //价格
    double g_price,total_price;
    g_price=0;
    total_price=0;
    /*******
    String sqlList="select * from orders where user_name='"+Session.getAttribute("username")+"'
and status=0";
    ResultSet RSList=workM.executeQuery(sqlList);
    try
    {
      while(RSList.next())
      {
        int b_num;
        b_num=RSList.getInt("book_number");
      }
    }
    %>
  <tr bgcolor="#CCFFCC">
    <td width="15%">
      <div align="center"> <font color="#0000FF">
        <input type="text" name="book_number" size="4" value=<%=b_num%>>
      </font></div>
    </td>
    <%

```

135

```

</td>
<td height="18" width="16%" bgcolor="#CCCC99">
    <div align="center"><font color="#0000FF">&yen;5.00</font></div>
</td>
</tr>
<tr bgcolor="#66FF66">
<td colspan="3">
    <div align="center"><b><font color="#FF0000">总费用</font></b></div>
</td>
<td width="16%">
    <div align="center"><b><font color="#FF0000">&yen;<%= (float)g_price+5%></font>
</b></div>
</td>
</tr>
<tr>
<td align="center">
<td colspan="4">
<input type="button" value="继续购物" onclick="javascript:self.close();">
<input type="submit" name="Submit" value="重新计算价格">
<input type="submit" name="Submit2" value="填写订单">
</td>
</tr>
</table>
</form>
<p>&nbsp;</p>
</div>
</body>
</html>

```

运行结果如图 5-5 所示。

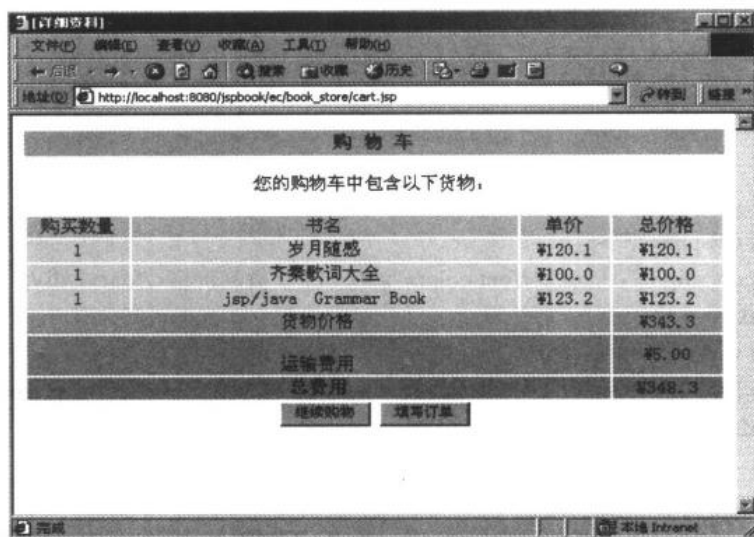


图 5-5 购物车效果图

## 第 6 章 JavaBeans 在 JSP 中的应用

在本章中，讲述了 JavaBeans 在 JSP 中的应用。首先讲述了 JavaBeans 的基本概念，接着讲述了 JavaBeans 的属性和方法，最后一节则通过实例讲述了 JavaBeans 在 JSP 中的应用。

通过对本章的学习，将会掌握 JavaBeans 在 JSP 开发中是如何得到应用的。

### 6.1 JavaBeans 的概念

JavaBeans 是 Sun 公司为进入因特网、企业网以及万维网上进行分布式计算的组件结构的入口。按照 Sun 公司的定义，JavaBeans 是一个可重复使用的软件部件，该部件可以用来生成其进行可视化处理的组件。

按照 JavaBeans 的说法，一个组件即 Java 应用程序或 Applet 的可重复使用的部件就是一个 Bean。

JavaBeans 体系结构是第一个全面地基于组件的标准模型之一。JavaBeans 是描述 Java 的软件组件模型，有点类似于 Microsoft 的 COM 组件概念。JavaBeans 组件是 Java 类，这些类遵循一个接口格式，以便于使方法命名、底层行为以及继承或实现的行为能够把类看做标准的 JavaBeans 组件的方式进行构造。

在 Java 模型中，通过 JavaBeans 可以无限扩充 Java 程序的功能，通过 JavaBeans 的组合可以快速生成新的应用程序。

JavaBeans 具有以下特性：

- 可以实现代码的重复利用
- 易维护性、易使用性、易编写性
- 可以在支持 Java 的任何平台上工作，而不需要重新编译
- 可以在内部、网内或者是网络之间进行传输
- 可以以其他部件的模式进行工作

JavaBeans 传统的应用在于可视化领域，如 AWT（窗口抽象工具集）下的应用。其实，基于 AWT 的任何 Java 程序已经是一个 Bean，完全可以把它作为一个组件来使用。当然，您也可以修改它的代码，来增加新功能，使之成为具有更多功能的 Bean 组件。

现在，JavaBeans 更多的应用在于不可视化领域，它在服务器端应用方面表现出了越来越强的生命力。不可视化的 JavaBeans 和可视化的 JavaBeans 同样使用属性和事件。不可视化的 JavaBeans 在 JSP 程序中常用来封装事务逻辑、数据库操作等，可以很好地实现业务逻辑和前台程序的分离，使得系统具有更好得健壮性和灵活性。

JavaBeans 描述了 JDK1.1 以前的 Java 所没有的东西——定制、交互部件。因此，运行 JavaBeans 最小的需求是 JDK1.1 或者以上的版本。



JavaBeans 最大的优点就是能够一次编写、多次使用，而且能够运行在任何 Java 虚拟机运行的地方，另外其代码相对来说也比较容易编写。

JavaBeans 是 ActiveX 的直接竞争对手，而上述几个方面 ActiveX 则逊色很多。

JavaBeans 是 Java 的组件模型。在 JavaBeans 规范中定义了事件和属性等特征。EJB (Enterprise JavaBeans) 也定义了一个 Java 组件模型，但是 Enterprise JavaBeans 组件模型和 JavaBeans 组件模型是不同的。

JavaBeans 允许开发者在开发工具中可视化地操作组件，JavaBeans 规范详细地解释了组件间事件的登记、传递、识别以及属性的使用、定制、应用接口等。而 Enterprise JavaBeans 的侧重点则是详细地定义了一个可以合理部署 Java 组件的服务框模型，并没有涉及到事件，因为 EJB 通常不会接受或者发送事件。

Enterprise JavaBeans 为开发服务器端应用程序组件提供了一个模型，利用这个模型来创建可移植与分布式企业应用程序服务器或组件。Enterprise JavaBeans 是一个用来创建分布式、服务器端以及基于 Java 的企业应用程序组件的功能强大的组件模型。

Enterprise JavaBeans 组件模型是 J2EE (Java 2 Enterprise Edition) 体系结构的核心部分。Enterprise JavaBeans 体系结构中的客户端包含了调用 EJB 中特定业务方法所需要的 EJB 接口，同时也包含了服务器端处理对象的管理句柄。在 EJB 体系结构的服务器端则包含有实际的 EJB 组件实现的实例，同时还包含用来调用在客户与 EJB 之间进行映射的容器代码。

## 6.2 JavaBeans 技术概述

### 6.2.1 JavaBeans 的属性

属性是 Bean 组件内部状态的抽象表示。JavaBeans 的属性可以分为以下四类：

- Simple 简单的
- Indexed 索引的
- Bound 绑定的
- Constrained 约束的

简单属性依赖于标准命名约定来定义 `getXXX()` 方法和 `setXXX()` 方法。索引属性则允许读取和设置整个数组，也允许使用数组索引单独地读取和设置数组元素。绑定属性则是其值发生变化时要广播给属性变化监听器的属性。约束属性则是那些值发生改变及起作用之前，必须由约束属性变化监听器生效的属性。

#### 1. 简单属性 (Simple)

一个简单属性表示一个伴随有 `get/set` 方法的变量。基本上，采用通常所说的存取方法来设置和检索 Bean 的属性。即：

```
public void set<PropertyName>(<PropertyType>value);
```

```
public <PropertyType> get<PropertyName>();
```

例如，对于 “MyBean” 属性，我们可以采用下面的方法：

```
public void setMyBean(MyBeanType value);  
public MyBeanType get MyBean();
```

对于布尔属性，我们使用下面的方法：

```
public boolean isMyBooleanBean();  
public void setMyBooleanBean(boolean value);
```

您可以设置的属性是读写、只读或者是只写。getXXX()方法和 setXXX()方法可以使之具有读写属性；getXXX()方法将允许您只具有只读属性；setXXX()方法将允许您只具有只写属性。如例程 6-1 所示。

例程 6-1

```
package input;  
/*文件名为 formInput.java*/  
public class formInput {  
    //类型为 String, 属性名为 str  
    String str = new String("NewYork Trade Center");  
    public formInput()  
    {  
    }  
    //set 属性  
    public void setInput(String str) {  
        this.str = str;  
    }  
    //get 属性  
    public String getInput() {  
        return str;  
    }  
}
```

## 2. 索引属性 (Indexed)

一个 Indexed 索引属性表示一个数组值。同上所述的 Simple 简单属性一样，可以使用 get/set 方法取得数组中的值。如例程 6-2 所示。

例程 6-2

```
package input1;  
/*文件名为 formInput1.java*/  
public class formInput1 {  
    //b 是一个 Indexed 索引的属性  
    String b[] = new String[] { "5", "2", "3", "4" };  
    public formInput1 ()  
    {  
    }  
    //取得整个数组的值  
    public String[] getInput() {  
        return b;  
    }  
    //设置整个数组
```

```
public void setInput(String [] b) {  
    this.b = b;  
}  
}
```

### 3. 绑定属性 (Bound)

绑定属性提供一种机制，即通知监听器一个 JavaBeans 组件的属性发生了改变。监听器实现了 `PropertyChangeListener` 接口并接收由 JavaBeans 组件产生的 `PropertyChangeEvent` 对象，`PropertyChangeEvent` 对象包括一个属性名字，旧的属性值以及每一个监听器可能要访问的新属性值。

JavaBeans 组件可以使用 `PropertyChangeSupport` 辅助程序类激活监听器要接收的事件。`PropertyChangeSupport` 对象使用一个指向 JavaBeans 组件实例的引用进行构造，并基于以下事实：

JavaBeans 实现了 `addPropertyChangeListener()` 和 `removePropertyChangeListener()` 方法以便加入和删除属性变化监听器。`PropertyChangeSupport.firePropertyChange()` 方法可以被使用，并传递属性名、旧值以及新值等信息。如例程 6-3 所示。

例程 6-3

```
/*文件名为 boundBean.java*/  
public class boundBean extends Canvas{  
    // str 是一个 bound 属性  
    String str= "bound Bean";  
    //实例化了一个 propertyChange 对象  
    private PropertyChangeSupport propertyChange = new PropertyChangeSupport(this);  
    public void setString(string newValue){  
        String oldValue= str;  
        str = newValue;  
        // str 的属性值已发生变化  
        propertyChange.firePropertyChange("change value", oldValue, newValue);  
    }  
    public String getString(){  
        return str;  
    }  
    public void addPropertyChangeListener(PropertyChangeListener propertyChangeListener)  
    {  
        propertyChange.addPropertyChangeListener(propertyChangeListener);  
    }  
    public void removePropertyChangeListener(PropertyChangeListener  
propertyChangeListener)  
    {  
        propertyChange.removePropertyChangeListener(propertyChangeListener);  
    }  
}
```

在上面的代码中，调用 `propertyChange` 的 `addPropertyChangeListener()` 方法把其他 JavaBeans 注册到 `str` 属性的监听器队列 `propertyChangeListener` 中，`propertyChangeListener`

本身是一个 Vector 数组，可以存储任何 Java 对象。

也可以调用 `propertyChange` 的 `removePropertyChangeListener()` 方法，从 `propertyChangeListener` 中注销指定的对象。

#### 4. 约束属性 (Constrained)

约束属性和绑定属性类似，但是属性值的变化首先要被所有的监听器验证之后，值的变化才能由 JavaBeans 组件发生作用。一个 JavaBeans 的约束属性是指当这个属性的值要发生变化时，与这个属性已经建立了某种连接的其他 Java 对象可否决属性值的改变。任何一个监听器都可以否决属性变化，JavaBeans 组件同意不使属性变化才能生效。

监听器实现了 `VetoableChangeListener` 接口并接收由 JavaBeans 组件产生的 `PropertyChangeEvent` 对象。JavaBeans 组件可以使用 `VetoableChangeSupport` 辅助程序类激活由监听器接收的实际事件。

使用一个 JavaBeans 组件实例的引用来构造 `VetoableChangeSupport` 对象，并且 JavaBeans 实现了用 `addVetoableChangeListener()` 方法和 `removeVetoableChangeListener()` 方法来加入或者删除可否决属性变化监听器。`VetoableChangeSupport.fireVetoableChange()` 方法可以用来传递属性的名字、旧属性值和新的属性值等信息。如例程 6-4 所示。

例程 6-4

```
/*文件名为 constrainedBean.java*/
public class constrainedBean {
    //实例化了一个 propertyChange 对象
    private PropertyChangeSupport propertyChange=new PropertyChangeSupport(this);
    //实例化了一个 vetoChange 对象
    private VetoableChangeSupport vetoChange=new VetoableChangeSupport(this);
    public void setUsername(String  newUsername)
    {
        try{

            String  oldUsername=yourName;
            vetoChange.fireVetoableChange("Username",    new    String(oldUsername),    new    String
(newUsername));
            yourName = newUsername;
            changes.firePropertyChange("Username", new String (oldPriceInCents), new String (newUsername));
        }
        catch(PropertyVetoException  e)
        {
            System.err.println(e.getMessage());
        }
    }

    public void addVetoableChangeListener(VetoableChangeListener  vetoableChangeListener)
    {
        vetos.addVetoableChangeListener(vetoableChangeListener);
    }
    public void removeVetoableChangeListener(VetoableChangeListener vetoableChangeListener){
        vetos.removeVetoableChangeListener(vetoableChangeListener);
    }
}
```

}

在上面的代码中，可以看出一个约束属性有两种监听器：属性变化监听器和否决属性变化监听器。否决属性变化监听器有相应的控制语句，在监听到有约束属性要发生改变时，在控制语句中就会判断是否要否决这个属性的改变。

## 6.2.2 JavaBeans 的方法

方法是处理事件的手段，而事件处理则是 JavaBeans 体系结构的核心之一。

JavaBeans 容器环境可以接收 JavaBeans 组件事件通知，并且，如果 JavaBeans 组件符合一些简单规则，就可以在设计时选择 JavaBeans 组件可以响应的事件。在 JavaBeans 组件上加入和删除方法必须以标准方式定义，以便分别加入和删除事件监听器。JavaBeans 容器能够加入或删除对事件监听器的引用，它使用允许容器与组件事件交互的 JavaBeans 组件。

JavaBeans 组件上的事件可以用 Bean 进行注册——如果它实现了一个 addXXXListener (XXXListener) 形式的方法，其中 XXX 是事件类型的名字。同样，Bean 如果实现了一个 removeXXXListener (XXXListener) 方法，事件就可以被注销。

最后说明一点，如果 JavaBeans 组件在一个时刻只允许一个监听器，addXXXListener (XXXListener) 方法应声明其产生 java.util.TooManyListenersException。

## 6.3 JavaBeans 的应用

### 6.3.1 在 JSP 中使用 JavaBeans

下面是在 JSP 页面中使用 JavaBeans 的一个很简单的例子。JSP 页面实现了动态显示输入值的功能。

JSP 页面的代码如例程 6-5 所示，其运行结果如图 6-1 所示。

例程 6-5

```
<%@ page contentType="text/html; charset=GBK" %>
<html>
<head>
<title>
sample1
</title>
</head>
<jsp:useBean id="sample1BeanId" scope="session" class="Bean.sample1Bean" />
<jsp:setProperty name="sample1BeanId" property="*" />
<body>
<h1>
A JSP & Bean Sample
```

```
</h1>
<form method="post">
<br>Enter new value : <input name="sample"><br>
<br><br>
<input type="submit" name="Submit" value="Submit">
<input type="reset" value="Reset">
<br>
Value of Bean property is :<jsp:getProperty name="sample1BeanId" property="sample" />
</form>
</body>
</html>
```

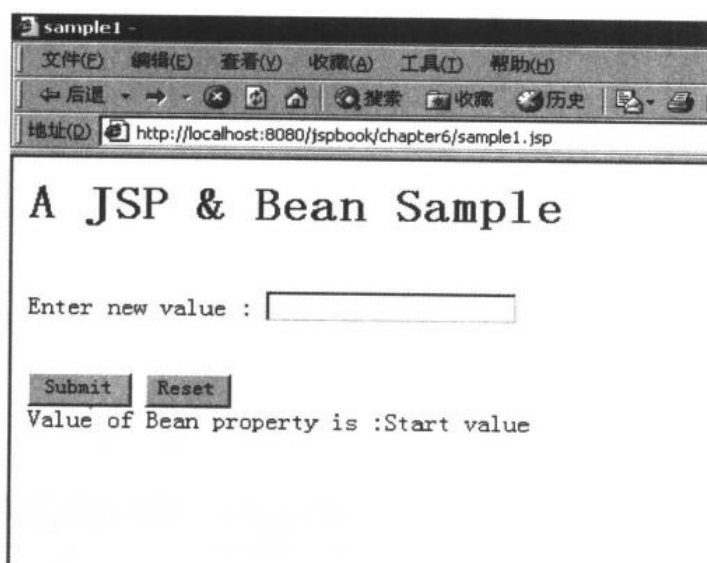


图 6-1 运行结果

JSP 页面所用到的 JavaBEAN 文件源代码如例程 6-6 所示，其运行结果如图 6-2 所示。

例程 6-6

```
package Bean;
public class sample1Bean {
    private String sample = "Start value";
    /**Access sample property*/
    public String getSample() {
        return sample;
    }
    /**Access sample property*/
    public void setSample(String newValue) {
        if (newValue!=null) {
            sample = newValue;
        }
    }
}
```



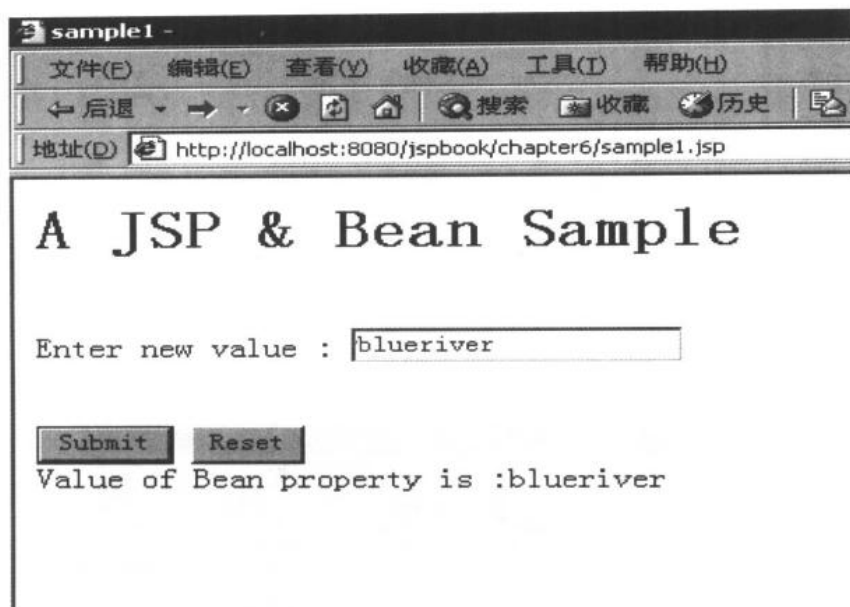


图 6-2 运行结果图

### 6.3.2 购物车范例

这是一个简单的购物车范例。选择物品放入购物车，运行效果如图 6-3 所示。



图 6-3 carts.html 运行结果图

选择物品放入购物车后，可以选择物品进行加入或者删除，运行效果如图 6-4 所示。  
Carts.jsp 的源代码如例程 6-7 所示。

例程 6-7

```
<html>
```

```
<jsp:useBean id="cart" scope="session" class="sessions.DummyCart" />
<jsp:setProperty name="cart" property="*" />
<%
    cart.processRequest(request);
%>
<FONT size = 5 COLOR="#CC0000">
<br> You have the following items in your cart:
<ol>
<%
    String[] items = cart.getItems();
    for (int i=0; i<items.length; i++) {
%>
<li> <%= items[i] %>
<%
    }
%>
</ol>
</FONT>
<hr>
<%@ include file ="/jsp/sessions/carts.html" %>
</html>
```

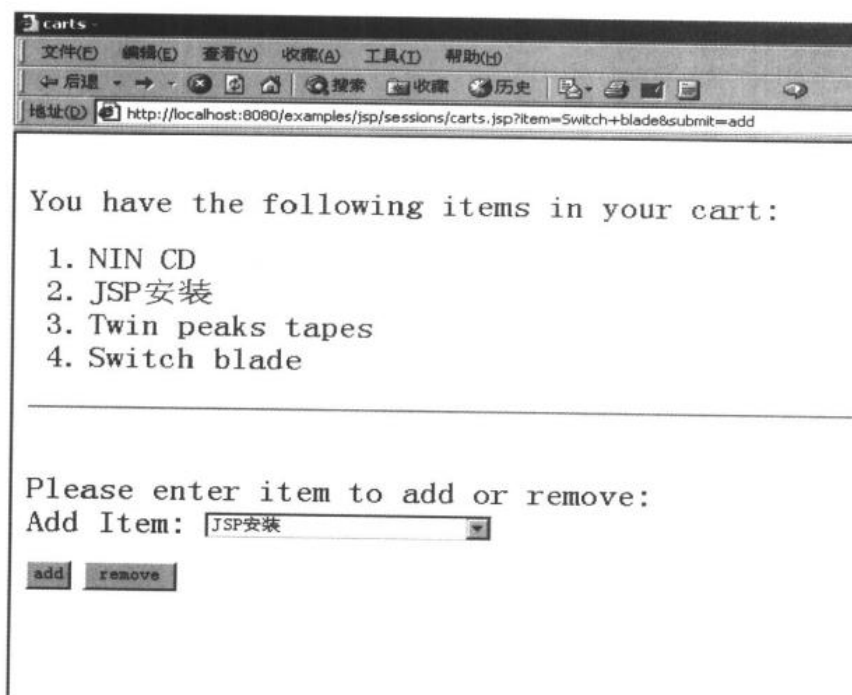


图 6-4 运行结果图

在该 JSP 页面中，起作用的 Bean 源代码如例程 6-8 所示。

例程 6-8

```
package sessions;
import javax.servlet.http.*;
import java.util.Vector;
import java.util.Enumeration;
public class DummyCart {
    Vector v = new Vector();
    String submit = null;
    String item = null;
    private void addItem(String name) {
        v.addElement(name);
    }
    private void removeItem(String name) {
        v.removeElement(name);
    }
    public void setItem(String name) {
        item = name;
    }
    public void setSubmit(String s) {
        submit = s;
    }
    public String[] getItems() {
        String[] s = new String[v.size()];
        v.copyInto(s);
        return s;
    }
    public void processRequest(HttpServletRequest request) {
        // null value for submit - user hit enter instead of clicking on
        // "add" or "remove"
        if (submit == null)
            addItem(item);

        if (submit.equals("add"))
            addItem(item);
        else if (submit.equals("remove"))
            removeItem(item);

        // reset at the end of the request
        reset();
    }
    // reset
    private void reset() {
        submit = null;
        item = null;
    }
}
```

这个 Bean 的作用范围是 Session，也就是在当前会话中起作用。

### 6.3.3 多选框范例

通过 Bean 实现显示多选的信息，图 6-5、图 6-6 是运行的效果图。

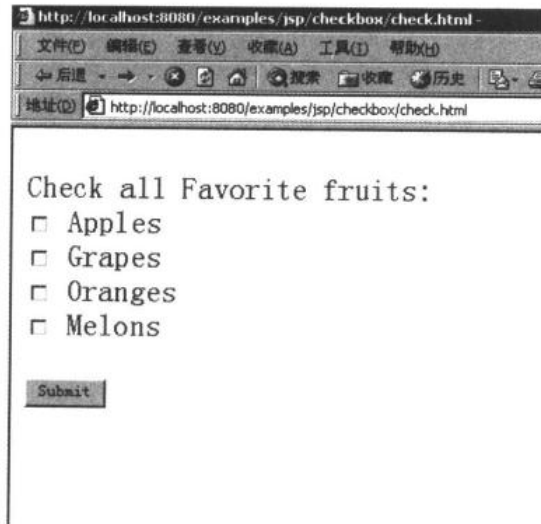


图 6-5 运行效果图 (1)

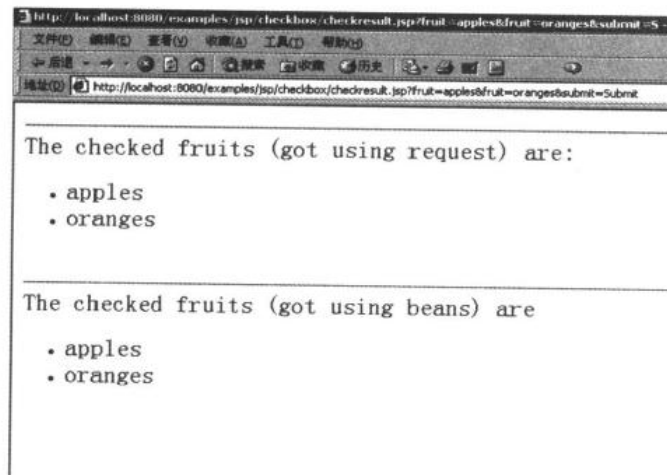


图 6-6 运行效果图 (2)

checkresult.jsp 源代码如例程 6-9 所示。

例程 6-9

```
<html>
<body bgcolor="white">
<font size=5 color="red">
<%! String[] fruits; %>
<jsp:useBean id="foo" scope="page" class="checkbox.CheckTest" />
```

```
<jsp:setProperty name="foo" property="fruit" param="fruit" />
<hr>
The checked fruits (got using request) are: <br>
<%
    fruits = request.getParameterValues("fruit");
%>
<ul>
<%
    if (fruits != null) {
        for (int i = 0; i < fruits.length; i++) {
%>
<li>
<%
            out.println (fruits[i]);
        }
    } else out.println ("none selected");
%>
</ul>
<br>
<hr>
The checked fruits (got using Beans) are <br>
<%
    fruits = foo.getFruit();
%>
<ul>
<%
    if (!fruits[0].equals("1")) {
        for (int i = 0; i < fruits.length; i++) {
%>
<li>
<%
            out.println (fruits[i]);
        }
    } else out.println ("none selected");
%>
</ul>
</font>
</body>
</html>
```

在该 JSP 页面中所用到的 Bean 源代码如例程 6-10 所示。

例程 6-10

```
package checkbox;

public class CheckTest {
```

```
String b[] = new String[] { "1", "2", "3", "4" };

public String[] getFruit() {
    return b;
}

public void setFruit(String [] b) {
    this.b = b;
}
}
```

这个 Bean 非常简单，可见在 JSP 中应该充分使用 Bean。

### 6.3.4 猜数字范例

这是一个猜数字的程序，程序指定一个 1~100 之间的某一个自然数，然后用户去猜这个数的值是多少。程序会根据用户的输入值，做出相应的信息提示。运行效果如图 6-7~图 6-9 所示。

numguess.jsp 源代码如例程 6-11 所示。

例程 6-11

```
<%@ page import = "num.NumberGuessBean" %>
<jsp:useBean id="numguess" class="num.NumberGuessBean" scope="session"/>
<jsp:setProperty name="numguess" property="*/"/>
<html>
<head><title>Number Guess</title></head>
<body bgcolor="white">
<font size=4>
<% if (numguess.getSuccess()) { %>
    Congratulations! You got it.
    And after just <%= numguess.getNumGuesses() %> tries.<p>
    <% numguess.reset(); %>
    Care to <a href="numguess.jsp">try again</a>?
<% } else if (numguess.getNumGuesses() == 0) { %>
    Welcome to the Number Guess game.<p>
    I'm thinking of a number between 1 and 100.<p>
    <form method=get>
    What's your guess? <input type=text name=guess>
    <input type=submit value="Submit">
    </form>
<% } else { %>
    Good guess, but nope. Try <b><%= numguess.getHint() %></b>.
    You have made <%= numguess.getNumGuesses() %> guesses.<p>
    I'm thinking of a number between 1 and 100.<p>
    <form method=get>
    What's your guess? <input type=text name=guess>
```



```
<input type=submit value="Submit">
</form>
<% } %>
</font>
</body>
</html>
```

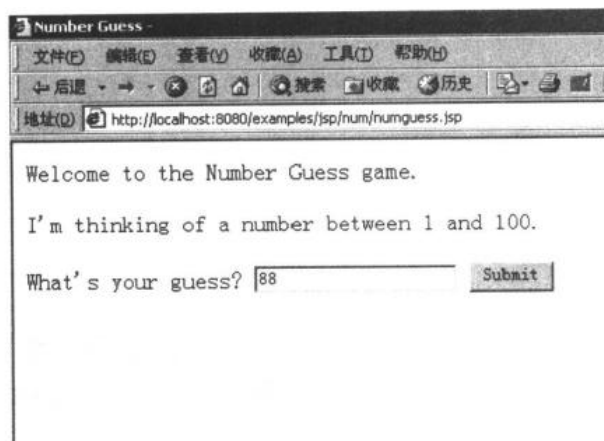


图 6-7 运行效果图 (1)

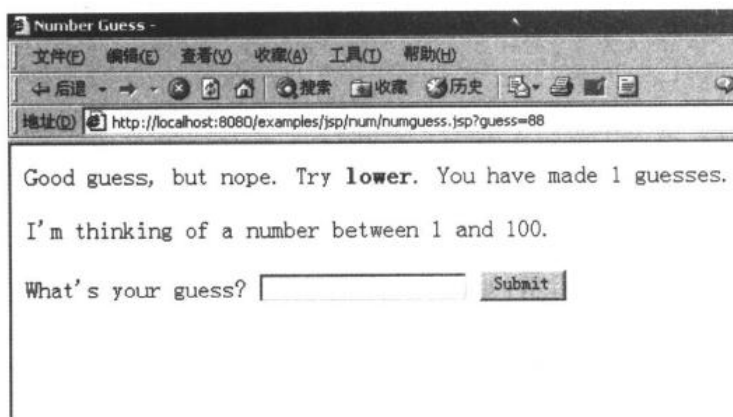


图 6-8 运行结果 (2)

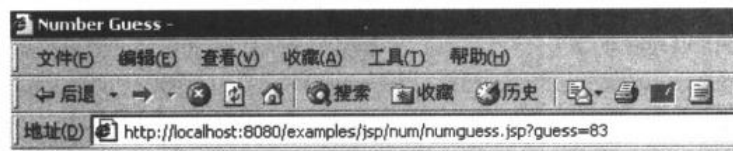


图 6-9 运行效果图 (3)

该 JSP 程序用到了一个 Bean，其源代码如例程 6-12 所示。

例程 6-12

```
package num;
import java.util.*;
public class NumberGuessBean {
    int answer;
    boolean success;
    String hint;
    int numGuesses;
    public NumberGuessBean() {
        reset();
    }
    public void setGuess(String guess) {
        numGuesses++;
        int g;
        try {
            g = Integer.parseInt(guess);
        }
        catch (NumberFormatException e) {
            g = -1;
        }
        if (g == answer) {
            success = true;
        }
        else if (g == -1) {
            hint = "a number next time";
        }
        else if (g < answer) {
            hint = "higher";
        }
        else if (g > answer) {
            hint = "lower";
        }
    }
    public boolean getSuccess() {
        return success;
    }
    public String getHint() {
        return "" + hint;
    }
    public int getNumGuesses() {
        return numGuesses;
    }
    public void reset() {
        answer = Math.abs(new Random().nextInt() % 100) + 1;
    }
}
```

```

        success = false;
        numGuesses = 0;
    }
}

```

### 6.3.5 封装数据库范例

下面介绍用 JavaBeans 封装数据库的操作，如果您还不熟悉数据库的知识，请参阅后面相关的章节。本例采用的数据库是 Microsoft SQL Server。首先，建立数据库 Vote，在其中建立表 voter 和 result。

表 voter 的作用是存放投票人的资料，其结构如表 6-1 所示，运行结果如图 6-10 所示。

表 6-1 投票人资料

字段名	描述
Candidate	投票的公司
voter_name	投票人性名
voter_company	投票人所在公司
voter_country	投票人所在国家
ip_address	投票人的 IP 地址
vote_time	投票时间

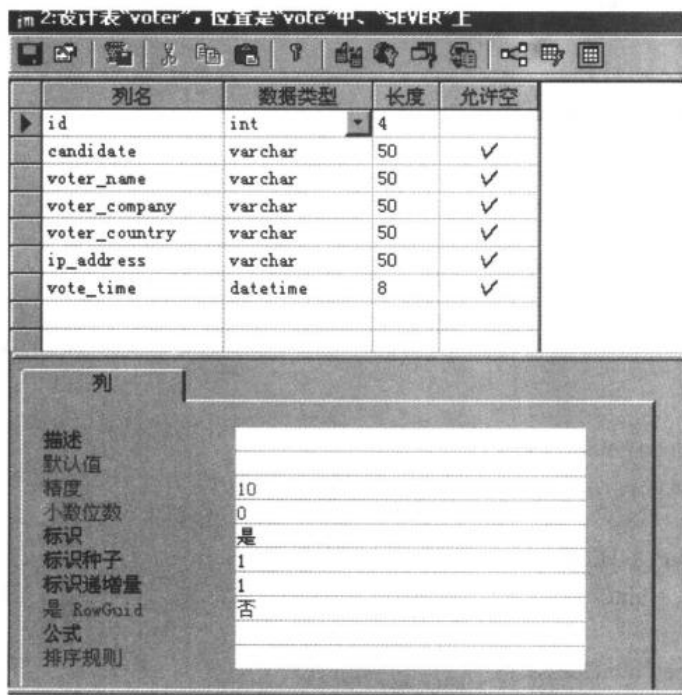


图 6-10 运行结果图

表 result 的作用是存放候选人的票数，其结构如表 6-2 所示，运行结果如图 6-11 所示。

表 6-2 候选人票数

字段名	描述
Candidate	候选公司
voter_num	票数

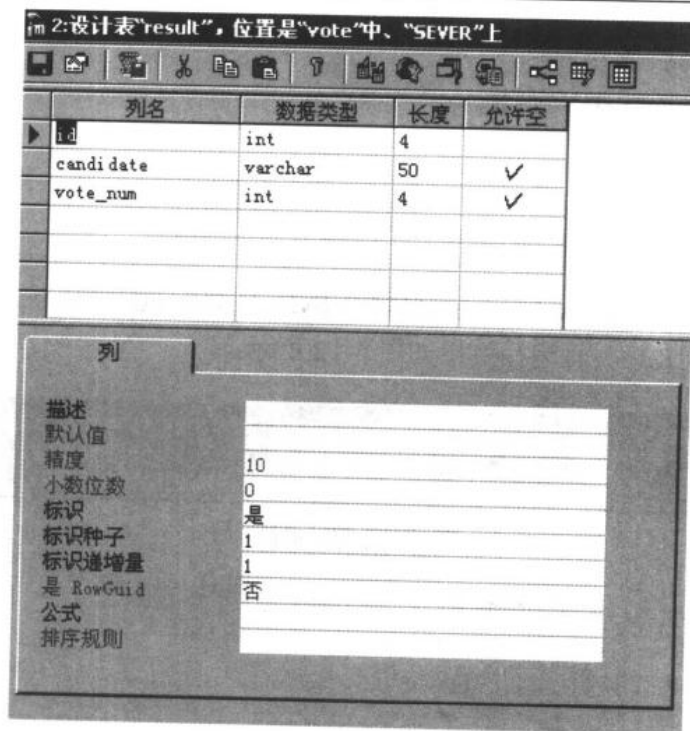


图 6-11 运行效果图

建立完数据库后，在 ODBC 数据源管理器中加入系统 DSN，取名为 vote。创建一个 JavaBeans，取名为 conn.java。源代码如例程 6-13 所示。

例程 6-13

```
package vote;
import java.sql.*;
public class conn {
    String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
    String sConnStr = "jdbc:odbc:vote";
    Connection connect = null;
    ResultSet rs = null;
    public conn() {
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

```
public ResultSet executeQuery(String sql) {  
    rs = null;  
    try {  
        connect = DriverManager.getConnection(sConnStr);  
        Statement stmt = connect.createStatement();  
        rs = stmt.executeQuery(sql);  
    }  
    catch(SQLException ex)  
    System.err.println(ex.getMessage());  
}  
return rs;  
}  
}
```

编译后放置在 D:\tomcat\webapps\test\WEB-INF\classes\vote 目录下。  
用来显示信息的 JSP 页面运行效果如图 6-12 所示。

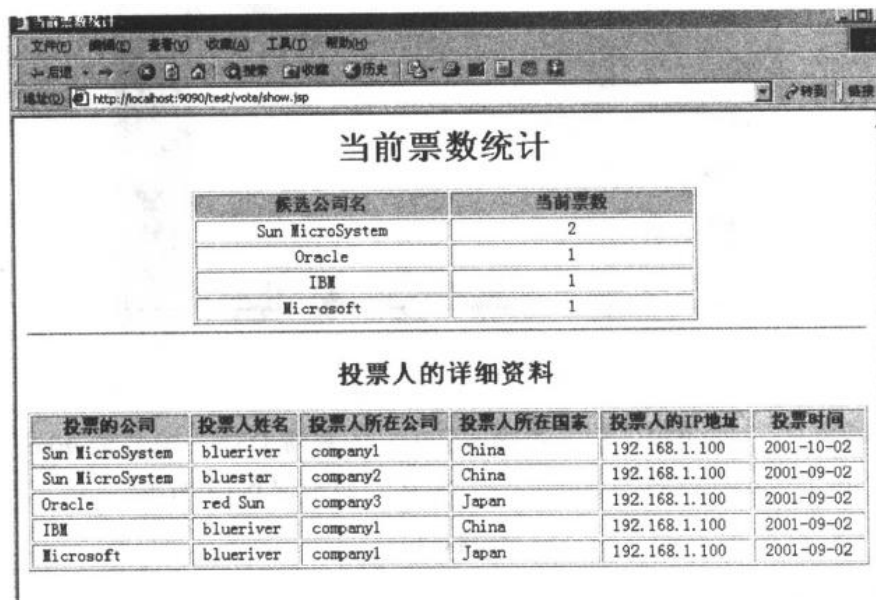


图 6-12 JSP 页面运行效果图

show.jsp 页面的源代码如例程 6-14 所示。

例程 6-14

```
<%@ page language="java" import="java.sql.*"%>  
<jsp:useBean id="voteBean" scope="page" class="vote.conn"/>  
<html>  
<head>  
<title>当前票数统计</title>  
<meta content="text/html; charset=gb_2312-80" http-equiv="Content-Type">  
<meta content="blueriver" name="Author">  
<meta http-equiv="refresh" content="10;URL=show.jsp"></head>
```

```

<body bgcolor="#FFFFFF">
<div align="center">
  <h1><b>当前票数统计</b> </h1>
  <table width="60%" border="1">
    <tr bgcolor="#CCCCFF">
      <td width="51%">
        <div align="center"><font color="#FF0033"><b>候选公司名</b></font></div>
      </td>
      <td width="49%">
        <div align="center"><font color="#FF0033"><b>当前票数</b></font></div>
      </td>
    </tr>
  </table>
  <%
  ResultSet RS_result;
  RS_result=voteBean.executeQuery("select * from result");
  String companyName;
  int voteNum;
  while(RS_result.next())
  {
  companyName=RS_result.getString("candidate");
  voteNum=RS_result.getInt("vote_num");
  %>
    <tr>
      <td width="51%" bgcolor="#FFFFFF">
        <div align="center"><%=companyName%></div>
      </td>
      <td width="49%">
        <div align="center"><%=voteNum%></div>
      </td>
    </tr>
  <%
  }
  RS_result.close();
  %>
  </table>
  <hr align="center">
  <h2><b>投票人的详细资料</b> </h2>
  <table width="100%" border="1">
    <tr bgcolor="#FFCCCC">
      <td>
        <div align="center"><font color="#0000FF"><b>投票的公司</b></font></div>
      </td>
      <td>
        <div align="center"><font color="#0000FF"><b>投票人姓名</b></font></div>
      </td>
    </tr>
  </table>

```



```

        <td>
            <div align="center"><font color="#0000FF"><b>投票人所在公司</b></font></div>
        </td>
        <td>
            <div align="center"><font color="#0000FF"><b>投票人所在国家</b></font></div>
        </td>
        <td>
            <div align="center"><font color="#0000FF"><b>投票人的 IP 地址</b></font></div>
        </td>
        <td>
            <div align="center"><font color="#0000FF"><b>投票时间</b></font></div>
        </td>
    </tr>
<%
    ResultSet RS_voter;
    RS_voter=voteBean.executeQuery("select * from voter");
    String candidate,voterName,voterCompany,voterCountry,ipAdress;
    java.util.Date voteTime;
    while(RS_voter.next())
    {
        candidate=RS_voter.getString("candidate");
        voterName=RS_voter.getString("voter_name");
        voterCompany=RS_voter.getString("voter_company");
        voterCountry=RS_voter.getString("voter_country");
        ipAdress=RS_voter.getString("ip_address");
        voteTime=RS_voter.getDate("vote_time");
    }
    %>
    <tr>
        <td>&nbsp;<%=candidate%></td>
        <td>&nbsp;<%=voterName%></td>
        <td>&nbsp;<%=voterCompany%></td>
        <td>&nbsp;<%=voterCountry%></td>
        <td>&nbsp;<%=ipAdress%></td>
        <td>&nbsp;<%=voteTime%></td>
    </tr>
<%
    }
    RS_voter.close();
    %>
</table>
<p>&nbsp;</p>
</div>
</body>
</html>

```

## 第 7 章 JSP 应用中的文件操作

在本章中，讲述了 JSP 中的文件操作。首先讲述了在 JSP 中是如何从文件中读取数据的，接着讲述了在 JSP 中是如何向文件中写入数据，以及在 JSP 中是如何向文件中追加数据的，最后一节则是一个构建计数器实例。

通过对本章的学习，将会掌握 Java 的 IO 操作在 JSP 开发中的应用。

在 Java 语言的的实际应用中，IO 操作是非常重要的一部分。同样，文件操作在 JSP 应用中也占据着同等重要的地位。

本章将从三个方面讲述 JSP 应用中的文件操作：从文件中读取数据、向文件中写入数据、向文件中追加数据。最后，将讲述一个实际的应用例子——构建计数器。

### 7.1 从文件中读取数据

下面，通过一个例子来讲述 JSP 是如何实现读取数据的，该例子是由一个 JSP 文件和 JavaBeans 文件实现的。

该例子中使用的 JSP 文件并不包含太多的 Java 代码，主要的代码放在 JavaBeans 中。由此我们也可以看到 JSP 和 JavaBeans 的基本联系。

在 JSP 文件中，首先要导入 JavaBeans，如：

```
<%@ page import="fileReader" %>
```

page 命令的意思是为整个 JSP 页面导入 JavaBeans。

告诉编译器将使用一个 JavaBeans，以及如何识别它，并进行初始化。

```
<JSP:useBean id="fileReader" class="fileReader" scope="page"/>
```

然后决定要设置那些属性。这里是"FileName"，因为我们要使用 Bean 的 setFileName 方法。所以 Bean 的名字必须包含：

```
<JSP:setProperty name="fileReader" property="FileName"/>
```

然后就展开实际的 HTML 页面。

```
<html>
```

```
<head><title>读取文件</title></head>
```

```
<body bgcolor="white">
```

现在开始编写一些 Java 脚本。首先检查文件名是否已经设置好，如果设置好了，就显示文件，否则我们要转到另一个页面。

```
<%if(fileReader.getFileName() != "") { %>
```

fileReader 是一个 JavaBeans，所以可以用 Java 类来存取它。现在得到文件名称：

```
文件名称是： '<% out.println(fileReader.getFileName()); %>'
```

如果文件内容不为空，则有：

```
<%if (fileReader.getContent() != null) { %>
```

可以建立一个 `textarea` (HTML) 并用 `getRows()` 和 `getColumns()` 方法来调节到合适的位置, 然后再将文件内容放入。

```
<Form>
<TEXTAREA rows = <% = fileReader. getRows() %> cols = <% = fileReader . getColumns()
%> id= textarea1 name= textarea1>
<%out.println(fileReader.getContent()); %>
</TEXTAREA>
</Form>
```

如果文件内容为空, 则会出现错误提示信息:

```
<% }else { %>
<% out.println(fileReader.getErrorMessage()); %>
<% } %>
```

重置所有值并返回主页:

```
<% fileReader.reset(); %>
<a href="fileReader.jsp">读取另外的文件</a>
<% }else { %>
```

如果文件名为空, 则显示错误信息页面。

#### 1. filereader.jsp 文件的源代码 (如例程 7-1 所示)

例程 7-1

```
<JSP:useBean id="file_reader" class="fileReader" scope="session"/>
<JSP:setProperty name="file_reader" property="fileName" />
<html>
<head><title>读取文件</title></head>
<body bgcolor="white">
<% if (file_reader.getFileName() != "") { %>
文件[ '<% out.println(file_reader.getFileName()); %>' ]内容 : <br><br>
<% if (file_reader.getContent() != null) { %>
<Form>
<TEXTAREA rows=<%= file_reader.getRows() %> cols=<%= file_reader.getColumns() %>
id=textarea1 name=textarea1>
<% out.println(file_reader.getContent()); %>
</TEXTAREA>
</Form>
<% } else { %>
<% out.println(file_reader.getErrorMessage()); %>
<% } %>
<br><br>
<% file_reader.reset(); %>
<a href="fileReader.jsp">读取另外的文件</a>
<% }
else
{ %>
读取文件内容的例子。
<form method=get>
```

```
输入文件名: <input type=text name=fileName>
<input type=submit value="读取文件内容">
</form>
<% } %>
</body>
</html>
```

## 2. 使用的 JavaBeans 文件源代码（如例程 7-2 所示）

例程 7-2

```
import java.io.*;
import java.awt.event.*;
import java.util.*;
/**
 * 该文件实现的 Bean 类是一个提供基本的读取文件功能的 Bean 类
 */
public class fileReader {
    private String fileName;
    private String errorMessage;
    private int columns;
    private int rowCount;
    /**
     * fileReader 的构造函数
     */
    public fileReader() {
        reset();
    }
    /**
     * 重置所有的变量
     */
    public void reset() {
        fileName = "";
        errorMessage = "";
        columns = 0;
        rowCount = 0;
    }
    /**
     * 设置错误信息
     */
    public void setErrorMessage(String errorMessage) {
        this.errorMessage = errorMessage;
    }
    /**
     * 返回错误信息
     */
    public String getErrorMessage() {
        return errorMessage;
    }
}
```

```
}  
/**  
 * 返回文件名  
 */  
public String getFileName() {  
    return fileName;  
}  
/**  
 * 设置文件名  
 */  
public void setFileName(String fileName) {  
    this.fileName = fileName;  
}  
/**  
 * 返回文件内容的行数  
 */  
public int getRows() {  
    return rowCount;  
}  
/**  
 * 返回文件中最大一行的列数  
 */  
public int getColumns() {  
    return columns;  
}  
/**  
 * 将文件内容返回到一个字符串中，  
 * 如果存在错误信息，则返回 null 值  
 */  
public String getContent() {  
    String content = "";  
    File file = new File(fileName);  
    if (!file.exists()) {  
        setErrorMessage("Error: The file '" + fileName + "' does not exists.");  
        return null;  
    }  
    else if (file != null) {  
        try {  
            BufferedReader reader = new BufferedReader(new FileReader(file));  
            String inLine = reader.readLine();  
            while (inLine != null) {  
                if (inLine.length() + 1 > columns)  
                    columns = inLine.length() + 1;  
                content += (inLine + System.getProperty("line.separator"));  
                inLine = reader.readLine();  
                rowCount++;  
            }  
        }  
    }  
}
```

```
}  
return content;  
}  
catch (IOException e) {  
    setErrorMessage("Error reading the file: " + e.getMessage());  
    return null;  
}  
}  
else {  
    setErrorMessage("Unknown error!");  
    return null;  
}  
}  
}
```

首次运行 filereader.jsp 文件，在输入框内输入文件名。则会出现如图 7-1 所示的界面。

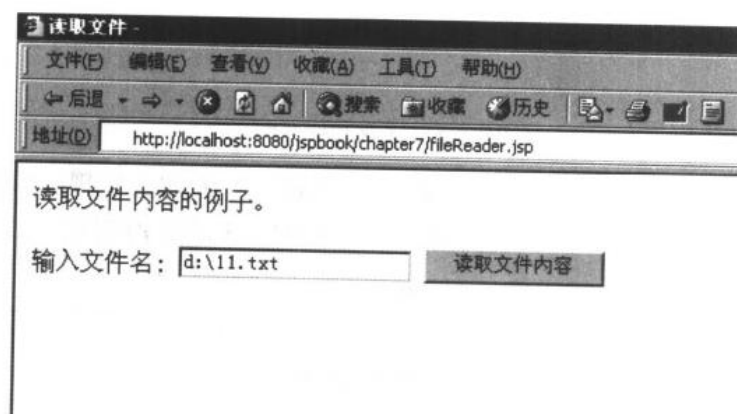


图 7-1 filertadev.jsp 文件运行界面

如果在 D 盘根目录下并不存在文件 11.txt，那么会出现如图 7-2 所示的界面。

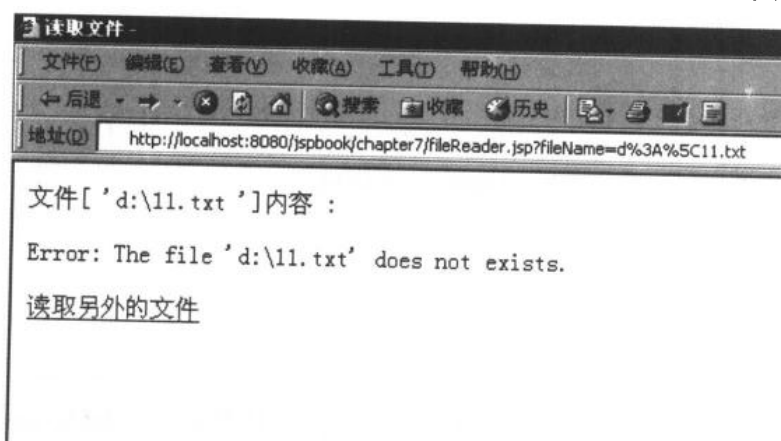


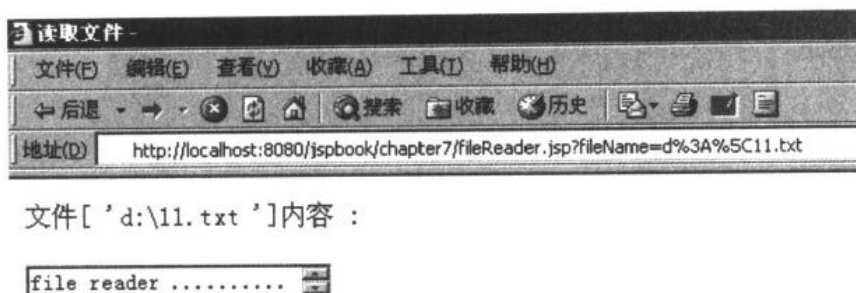
图 7-2 运行出错的界面



在 D 盘根目录下创建文件 11.txt，内容为：

file reader .....

单击【读取文件内容】按钮，则会出现如图 7-3 所示的界面。



读取另外的文件

图 7-3 读取文件界面

另外，值得一提的是在 Java 中写文件时，推荐用流的方式实现。因此，在 JSP 中从文件中读取数据时，也可以通过流的方式进行。例程 7-3 就能说明这一点。

例程 7-3

```
<%@ page language = "java" %>
<%@ page contentType = "text/html; charset=gb2312" %>
<%@ page import = "java.util.*" %>
<%@ page import = "java.lang.*" %>
<%@ page import = "javax.servlet.*" %>
<%@ page import = "javax.servlet.jsp.*" %>
<%@ page import = "javax.servlet.http.*" %>
<%@ page import = "java.io.*" %>
<%
int count=0;
FileInputStream fi =new FileInputStream ("/count.txt");
ObjectInputStream si= new ObjectInputStream (fi);
count =si.readInt();
count++;
out.print(count);
si.close();
%>
```

在该文件中，除了要导入几个包外，主要的是使用流：FileOutputStream、ObjectOutputStream。

## 7.2 向文件中写入数据

下面,通过例程 7-4 来讲述 JSP 是如何实现向文件中写入数据的,该例程也是由一个 JSP 文件和 JavaBeans 文件实现的。

### 1. 编程思路

在 JSP 文件中,首先要调用该例程中的 JavaBeans 文件。

告诉编译器将使用一个 JavaBeans,以及如何识别它,并进行初始化。

```
<JSP:useBean id="writer" class="fileWriter" scope="page">
```

然后决定要设置那些属性:

```
<JSP:setProperty name="writer" property="filePath" value="writer.txt" />
```

```
<JSP:setProperty name="writer" property="str" value="初始化内容" />
```

```
</JSP:useBean>
```

filePath 是指文件路径, str 是指写入的字符串。

调用 JavaBeans 中的 setStr 方法, 设置要写入的字符串:

```
<% writer.setStr("This is a string"); %>
```

调用 JavaBeans 中的 getStr 方法, 读取上面设置的字符串:

```
<% out.println(writer.getStr()); %><br>
```

调用 JavaBeans 中的 writeStr 方法写入文件并返回成功或者出错信息:

```
<% String resultMsg;  
resultMsg=writer.writeStr();  
if(resultMsg.equals("ok"))  
{  
    out.println("成功写入信息。");  
}  
else  
{  
    out.println(resultMsg);  
}  
%>
```

### 2. fileWriter.jsp 文件的源代码 (如例程 7-4 所示)

例程 7-4

```
<%--创建 javaBean 并设置属性 --%>  
<JSP:useBean id="writer" class="fileWriter" scope="page">  
<JSP:setProperty name="writer" property="filePath" value="writer.txt" />  
<JSP:setProperty name="writer" property="str" value="初始化内容" />  
</JSP:useBean>  
<html>  
<head>  
<title>写入数据</title>  
</head>  
<body>
```

```
<h3>写一个文件</h3>
<p>
<%--设置要写入的字符串 --%>
<% writer.setStr("This is a string"); %>
<%--读取上面设置的字符串 --%>
<% out.println(writer.getStr()); %><br>
<%--调用 writer 的 writeStr 方法写入文件并返回成功或者出错信息 --%>
<% String resultMsg;
resultMsg=writer.writeStr();
if(resultMsg.equals("ok"))
{
    out.println("成功写入信息。");
}
else
{
    out.println(resultMsg);
}
%>
</p>
</body>
</html>
```

### 3. 使用的 JavaBeans 文件源代码（如例程 7-5 所示）

例程 7-5

```
import java.io.*;
public class fileWriter
{
    //文件路径
    private String filePath;
    //写入的字符串
    private String str;
    //初始化
    public fileWriter()
    {
        filePath = null;
        str = "This is a string";
    }
    //设置文件路径
    public void setFilePath(String filePath)
    {
        this.filePath = filePath;
    }
    //得到文件路径
    public String getFilePath()
    {
        return filePath;
    }
}
```

```
}  
//得到字符串  
public void setStr(String str)  
{  
    this.str = str;  
}  
//设置字符串  
public String getStr()  
{  
    return str;  
}  
//写入字符串到文件中，成功则返回 ok 字符串  
public String writeStr()  
{  
    try  
    {  
        File f = new File(filePath);  
        PrintWriter out = new PrintWriter(new FileWriter(f));  
        out.print(this.getStr() + "\n");  
        out.close();  
        return "ok";  
    }  
    catch (IOException e)  
    {  
        return e.toString();  
    }  
}
```

执行写入数据的文件 `fileWriter` 运行效果如图 7-4 所示。

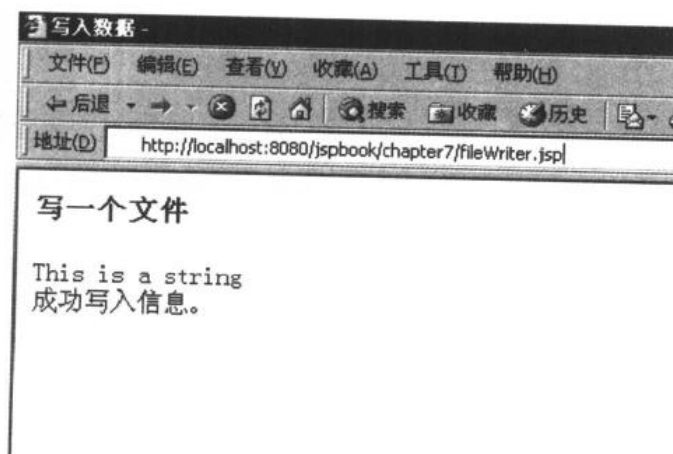


图 7-4 运行效果图

另外，值得一提的是在 Java 对写文件时，推荐用流的方式实现。因此，在 JSP 向文

件中写入数据时,也可以通过流的方式进行。例程 7-6 就能说明这一点。

该例程只有一个 JSP 文件。

例程 7-6

```
<%@ page language = "java" %>
<%@ page contentType = "text/html; charset=gb2312" %>
<%@ page import = "java.util.*" %>
<%@ page import = "java.lang.*" %>
<%@ page import = "java.io.*" %>

<%
int num;
num=100;
try
{
    FileOutputStream fo = new FileOutputStream ("/count.txt");
    ObjectOutputStream so = new ObjectOutputStream (fo);
    so.writeInt(num);
    so.close();
    out.println("成功写入数据");
}
catch(Exception e)
{
    out.println(e.toString());
}
%>
```

在该文件中,除了要导入几个包外,主要的是使用流: `FileOutputStream`、`ObjectOutputStream`。

运行效果图如 7-5 所示。

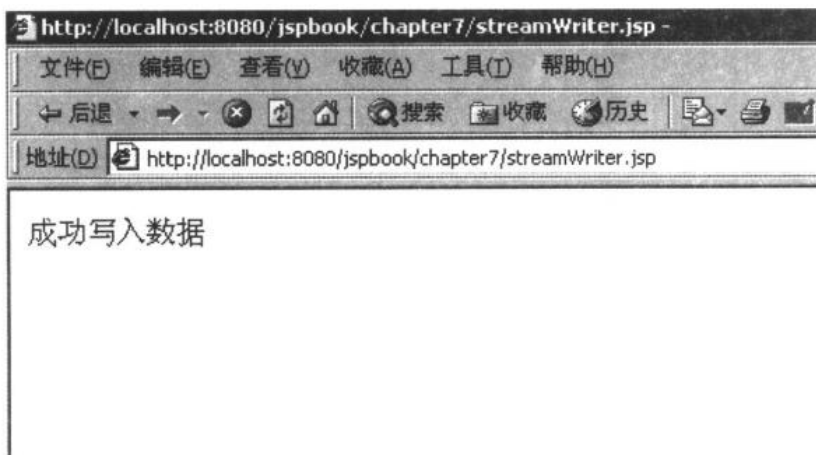


图 7-5 运行效果图

## 7.3 向文件中追加数据

下面,通过例程 7-7 来讲述 JSP 是如何实现向文件中追加数据的,该例子也是由一个 JSP 文件和 JavaBeans 文件实现的。

### 1. 编程思路

在 JSP 文件中,首先要调用该例子中的 JavaBeans 文件。

告诉编译器将使用一个 JavaBeans,以及如何识别它,并进行初始化。

```
<JSP:useBean id="append" class="WriteAppend" scope="page">
```

然后决定要设置那些属性:

```
<JSP:setProperty name="append" property="filePath" value="writer.txt" />
```

```
<JSP:setProperty name="append" property="str" value="初始化" />
```

```
</JSP:useBean>
```

filePath 是指文件路径, str 是指写入的字符串。

调用 JavaBeans 中的 setStr 方法, 设置要写入的字符串:

```
<% append.setStr("append"); %>
```

调用 JavaBeans 中的 getStr 方法, 读取上面设置的字符串:

```
<% out.println(append.getStr()); %>
```

调用 JavaBeans 中的 writeStr 方法写入文件并返回成功或者出错信息:

```
<% String resultMsg;  
resultMsg=append.writeStr();  
if(resultMsg.equals("ok"))  
{  
    out.println("成功写入信息。");  
}  
else  
{  
    out.println(resultMsg);  
}  
%>
```

### 2. writerAppend.jsp 文件的源代码

例程 7-7

```
<!--创建 javaBean 并设置属性 --%>  
<JSP:useBean id="append" class="WriteAppend" scope="page">  
<JSP:setProperty name="append" property="filePath" value="writer.txt" />  
<JSP:setProperty name="append" property="str" value="初始化" />  
</JSP:useBean>  
<html>  
<head>  
<title>追加数据</title>  
</head>  
<body>
```



```
<h3>追加数据</h3>
<p>
<%--设置要追加的字符串 --%>
<% append.setStr("aaaaaaaaaaaaaaaaappend"); %>
<%--读取上面设置的字符串 --%>
<% out.println(append.getStr()); %><br><br>
<%--调用 append 的 writeStr 方法追加文件并返回成功或者出错信息 --%>
<% String resultMsg;
resultMsg=append.writeStr();
if(resultMsg.equals("ok"))
{
    out.println("成功写入信息。");
}
else
{
    out.println(resultMsg);
}
%>
</p>
</body>
</html>
```

### 3. 使用的 JavaBeans 文件源代码（如例程 7-8 所示）

例程 7-8

```
import java.io.*;
public class WriteAppend
{
    //文件路径
    private String filePath;
    //追加的字符串变量
    private String str;
    //初始化
    public WriteAppend()
    {
        filePath = null;
        str = "Default message";
    }
    //设置文件路径
    public void setFilePath(String filePath)
    {
        this.filePath = filePath;
    }
    //得到文件路径
    public String getFilePath()
    {
        return this.filePath;
    }
}
```

```
}  
//设置要追加的字符串  
public void setStr(String str)  
{  
    this.str = str;  
}  
//得到要追加的字符串  
public String getStr()  
{  
    return this.str;  
}  
//追加字符串  
public String writeStr()  
{  
    try  
    {  
        //创建文件 filePath 并写入 str 字符串，注意和写入篇的区别  
        FileWriter f = new FileWriter(filePath,true);  
        PrintWriter out = new PrintWriter(f);  
        out.print(str + "\n");  
        out.close();  
        //关闭文件并返回 success 字符串  
        f.close();  
        return "ok";  
    }  
    catch (IOException e)  
    {  
        return e.toString();  
    }  
}  
}
```

执行追加数据的文件 `writerAppend` 运行效果如图 7-6 所示。

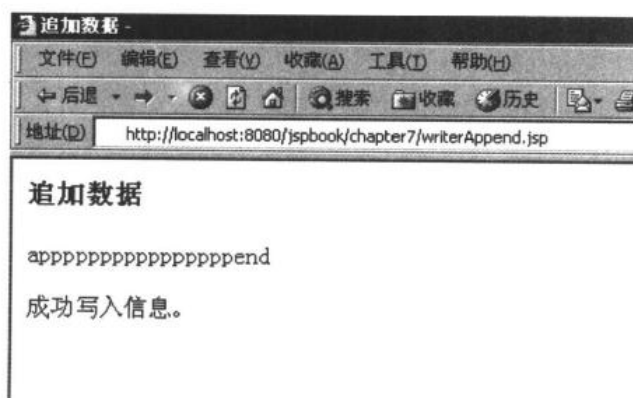


图 7-6 运行效果图

## 7.4 构建计数器实例

在 JSP 实际的开发中, 文件操作可以很方便地用来处理一些事务。譬如说, 我们完全可以使用前面几节的知识, 来构建一个计数器实例。当然, 文件操作的作用绝对不仅仅局限于此。

### 1. 编程思路

在该实例中, 用到了一个 JSP 文件和一个 JavaBeans 文件。

在 JSP 文件中, 首先要调用该例子中的 JavaBeans 文件。

```
<JSP:useBean id="counter" class="counter" scope="request">
</JSP:useBean>
```

获取文件 counter.txt 的绝对路径:

```
txtPath=request.getRealPath("counter.txt");
```

调用 counter 对象的 ReadFile 方法来读取文件 counter.txt 中的计数:

```
String count=counter.ReadFile(txtPath);
```

实现同步, 以便避免多个用户同时请求时, 导致技术不准确:

```
synchronized(txtPath)
```

调用 counter 对象的 ReadFile 方法来将计数器加一后写入到文件 counter.txt 中:

```
counter.WriteFile(txtPath,count);
```

### 2. counter.jsp 文件的源代码 (如例程 7-9 所示)

例程 7-9

```
<%@ page contentType="text/html; charset=gb2312"%>
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<TITLE>
//计数器演示程序
</TITLE>
</HEAD>
<BODY>
<!--创建并调用 Bean(counter)-->
<JSP:useBean id="counter" class="counter" scope="request">
</JSP:useBean>
<%
String txtPath;
//获取绝对路径
txtPath=request.getRealPath("counter.txt");
//调用 counter 对象的 ReadFile 方法来读取文件 counter.txt 中的计数
String count=counter.ReadFile(txtPath);
//实现同步
synchronized(txtPath)
{
```

//调用 counter 对象的 ReadFile 方法来将计数器加一后写入到文件 counter.txt 中  
counter.WriteFile(txtPath,count);

}

%>

您是第<font color="red"><b><%=count%></b></font>位访问者

</BODY>

</HTML>

用到的 JavaBeans 文件源代码为:

import java.io.\*;

public class counter

{

//保存文本的变量

private String currentRecord = null;

//BufferedReader 对象, 用于读取文件数据

private BufferedReader file;

//文件完整路径名

private String path;

public counter() {

}

//ReadFile 方法用来读取文件 filePath 中的数据, 并返回这个数据

public String ReadFile(String filePath) throws FileNotFoundException

{

path = filePath;

//创建新的 BufferedReader 对象

file = new BufferedReader(new FileReader(path));

String returnStr =null;

try

{

//读取一行数据并保存到 currentRecord 变量中

currentRecord = file.readLine();

}

catch (IOException e)

{

//错误处理

System.out.println("读取数据错误.");

}

if (currentRecord == null)

{

//如果文件为空

returnStr = "没有任何记录";

}

else

{

//文件不为空

returnStr =currentRecord;

```
}  
//返回读取文件的数据  
return returnStr;  
}  
//ReadFile 方法用来将数据 counter+1 后写入到文本文件 filePath 中  
//以实现计数增长的功能  
public void WriteFile(String filePath,String counter) throws FileNotFoundException  
{  
    path = filePath;  
    //将 counter 转换为 int 类型并加一  
    int Writestr = Integer.parseInt(counter)+1;  
    try  
    {  
        //创建 PrintWriter 对象，用于写入数据到文件中  
        PrintWriter pw = new PrintWriter(new FileOutputStream(filePath));  
        //用文本格式打印整数 Writestr  
        pw.println(Writestr);  
        //清除 PrintWriter 对象  
        pw.close();  
    }  
    catch(IOException e)  
    {  
        //错误处理  
        System.out.println(e.getMessage());  
    }  
}
```

在响应的目录下建立一个 counter.txt 文件，文件内容可以初始化为 0。  
执行效果可能如图 7-7 所示。



图 7-7 执行效果图

到此为止，JSP 应用中的文件操作以及实例应用已经全部讲述完毕。但是，文件操作的作用绝不仅仅只有这些。有兴趣的读者可以专门阅读讲述 IO 数据的书籍。

通过对本章的学习，将会掌握 Java 的 IO 操作在 JSP 开发中是如何得到应用的。

在下一篇中，将会讲述 Servlet 技术在 JSP 开发中的应用，包括 Servlet 技术的一些基本原理和常用类接口的介绍以及 JSP 与 Servlet 的结合编程。





# 开发专家之

# Sun ONE

## 第三篇 Servlet 技术的应用

在本篇中，将讲述 Servlet 技术在 JSP 开发中的应用。

本篇分为两章，分别讲述了 Servlet 技术的一些基本原理和常用类接口，以及 JSP 与 Servlet 的结合编程。

在第 8 章中，讲述了 Servlet 技术的基本概念、原理以及 Servlet 常用类接口等。首先讲述了 Servlet 技术的工作原理，接着介绍了 Servlet 与 JSP 之间的关系、Servlet 应用范围及其缺陷、Servlet 的生命周期以及 Servlet 常用类接口，最后一节则是介绍了 JSP 内建对象与 Servlet 中类的对应关系。

在第 9 章中，讲述了 Servlet 技术与 JSP 技术是如何结合使用的。首先讲述了 JSP 技术在网站开发中的两种模式，接着讲述了这两种模式的实例。在 9.3 和 9.4 节中则讲述了在 JSP 中是如何发送邮件的。

通过对本篇的学习，将会掌握 Servlet 技术与 JSP 技术在应用开发中的结合使用。



# 第 8 章 Servlet 技术

在本章中，讲述了 Servlet 技术的基本概念、原理以及 Servlet 常用类接口等。首先讲述了 Servlet 技术的工作原理，接着介绍了 Servlet 与 JSP 之间的关系、Servlet 应用范围及其缺陷、Servlet 的生命周期以及 Servlet 常用类接口（包括 `HttpServlet`、`HttpServletRequest`、`HttpServletResponse`、`HttpSession`、`ServletConfig`、`ServletContext`），最后一节则是介绍了 JSP 内建对象与 Servlet 中类的对应关系。

通过对本章的学习，将会掌握 Servlet 技术在 JSP 开发中是如何得到应用的。

## 8.1 Servlet 技术概述

### 8.1.1 什么是 Servlet

Servlet（Java 服务器小程序）是用 Java 编写的服务器端程序，是由服务器端调用和执行的任何 Java 类。Servlet 是使用 Java Servlet 应用程序设计接口（API）及相关类和方法的 Java 程序。除了 Java Servlet API，Servlet 还可以使用用以扩展和添加到 API 的 Java 类软件包。

Servlet 是 Java 2.0 中新增的一个全新功能，是一种采用 Java 技术来实现 CGI 功能的一种技术。Servlet 本身与协议无关，与平台也无关。浏览器端也就是客户端运行的 Java 程序叫做 Applet，而服务器端运行的 Java 程序叫做 Servlet。

Java Servlet 运行于 Java-enabled Web 服务器中，可以被看做是运行在请求或面向请求服务器上的模块。Java Servlet 能够像 CGI 脚本一样动态地扩展 Web 服务器的功能，并采用请求——响应模式提供 Web 服务。Servlet 和 CGI 一样都是运行在 Web 服务器上，用来生成 Web 页面的。

最早支持 Servlet 技术的是 JavaSoft 的 Java Web Server。此后，一些其他的基于 Java 的 Web Server 开始支持标准的 Servlet API。Servlet API 是用来编写 Servlet 的一套编程接口，编写 Servlet 不需要关心 Servlet 是如何被装载、Servlet 运行的服务器环境是什么、传输数据的协议有什么不同等。因而，Servlet 能够运行在不同的 Web 服务器中，Servlet 避免了 CGI 的这些缺陷。

Servlet 看起来像是通常的 Java 程序。Servlet 导入特定的属于 Java Servlet API 的包。因为是对象字节码，可动态地从网络加载，可以说 Servlet 对 Server 就如同 Applet 对 Client 一样，但是，由于 Servlet 运行于 Server 中，它们并不需要一个图形用户界面。

Servlet 的主要功能在于交互式地浏览和修改数据，生成动态 Web 内容。这个过程为：

- 客户端发送请求至服务器端。

- 服务器将请求信息发送至 Servlet。
- Servlet 生成响应内容并将其传给 Server。响应内容动态生成，通常取决于客户端的请求。
- 服务器将响应返回给客户端。

### 8.1.2 Servlet 与其他开发技术 (CGI - BIN) 的比较

概括来讲，Servlet 可以完成和 CGI 相同的功能。

Servlet 提供了 Java 应用程序的所有优势——可移植、稳健、易开发。使用 Servlet Tag 技术，Servlet 能够生成嵌于静态 HTML 页面中的动态内容。

Servlet 对 CGI 的最主要优势在于一个 Servlet 被客户端发送的第一个请求激活，然后它将继续运行于后台，等待以后的请求。每个请求将生成一个新的线程，而不是一个完整的进程。多个客户能够在同一个进程中同时得到服务。一般来说，Servlet 进程只是在 Web Server 卸载时被卸载。

CGI 应用开发比较困难，因为它要求程序员有处理参数传递的知识，这不是一种通用的技能。CGI 不可移植，为某一特定平台编写的 CGI 应用只能运行于这一环境中。每一个 CGI 应用存在于一个由客户端请求激活的进程中，并且在请求被服务后被卸载。这种模式将引起很高的内存、CPU 开销，而且在同一进程中不能服务多个客户。

Java Servlet 比传统的 CGI 技术来说，效率更高，使用更方便，功能更强大，更小巧也更便宜。

#### 1. Java Servlet 技术要比传统的 CGI 技术效率更高

传统的 CGI 对每一个 HTTP 请求都要产生一个新的进程。如果某个 CGI 程序是一个执行非常快的操作，打开进程的时间也许占用了大部分执行时间。而对 Java Servlet 来说，在 Java 虚拟机上，每一个请求由一个 Java 线程(thread)响应，而不是一个操作系统进程。类似地，传统 CGI 如果对同一个 CGI 程序有 N 个同时请求，这个 CGI 程序的代码将被导入内存 N 次。Java Servlet 产生 N 个线程，但只有一个 Java Servlet 类。Servlet 存储在内存中，运行速度非常快。同时 Java Servlet 比 CGI 有更多的优化选择，像预先计算，打开数据库连接等。

#### 2. Java Servlet 技术要比传统的 CGI 技术功能强大

Java Servlet 能够方便的处理 HTML 表单数据，也能够读取和设置 HTTP 头信息，并且能够处理 Cookies、跟踪 Sessions 等其他大量功能。

Java Servlet 可以很容易地实现对 CGI 来说是不可能或很困难的事务。例如：Java Servlet 能直接和服务器进行通信而 CGI 是不能的。

#### 3. Servlet 是模块化的

每一个 Servlet 可以执行一个特定任务，并且可以将它们并在一起工作。Servlet 之间是可以相互交流的。

#### 4. Java Servlet 之间能共享数据

它能方便地实现管理从 Request 到 Request 的请求，简化 Session 和获取前一页面。而 CGI 之间通信差，由于每个 CGI 程序的调用都开始一个新的进程，调用间通信通常要通



过文件进行，因而相当缓慢。同一台服务器上的不同 CGI 程序之间的通信也相当麻烦。

#### 5. Java Servlet 技术要比传统的 CGI 技术的调用时间要短得多

CGI 程序是作为单独过程运行的，通常调用时间较长，这个间接成本在每次调用的时候都要发生。在使用解释器的时候调用时间会更长。而内存中的 Servlet 可以非常迅速地加载。因此，Java Servlet 技术要比传统的 CGI 技术在调用时间上短得多。

#### 6. Java Servlet 技术要比传统的 CGI 技术安全

有些 CGI 版本有明显的安全弱点。即使是使用最新的标准和 PERL 等语言，系统也没有基本安全框架，而要靠一组事实上的规则。而 Java 定义有完整的安全机制，包括 SSL, CA 认证、安全政策等规范。

#### 7. Java Servlet 技术具有 Java 技术的所有优点

Servlet 是用 Java 编写的，Servlet 具有 Java 技术的几乎所有的优点，如可移植性、稳健性以及易开发、易维护等特性。

下面，我们通过一个例子来比较一下 Java Servlet 技术与传统的 CGI 技术之间的区别。

Java Servlet 技术与传统的 CGI 技术两者要实现相同的功能：获得客户端在浏览器表单中输入的参数 (UserName, UserPassword)，并将结果返回给客户端浏览器。相应的 CGI 程序和 Servlet 程序分别如例程 8-1 和例程 8-2 所示。

例程 8-1

```
$ query-string= $ ENV{?QUERY-STRING?};
$ query-string=As/%([dA-Fa-f][dA-Fa-f])/pack(" C" ,hex() $ 1))/eg; $ query-string=As/
+//g;
@pairs=split(/&/, $ query-string) ;
foreach $ pair(@pairs)
( $ key, $ value ) =split(/=/, $ pair) ;
$ form-data{ $ key}= $ value;
}
$ UserName = $ form-data{ " UserName " };

$ UserPassword = $ form-data{ " UserPassword " };
print " 您输入的用户名是 $ UserName , 输入的密码是 $ UserPassword. " ;
```

例程 8-2

```
import Java.io.*;
import Java.Servlet.*;
import Javax.Servlet.*;
public class test extends HttpServlet
{
    public void service(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {
        ServletOutputStream out=response.getOutputStream();
        response.setContentType(" text/html " );
    }
}
```



```
String UserName =request.getParameter(" UserName" );
String UserPassword =request.getParameter(" UserPassword" );
out.println(" 您输入的用户名是" +UserName  + " , 输入的密码是" );
out.println(UserPassword+ " 。 " );
out.close()
}
}
```

可以很明显地看到 Java Servlet 源代码比相应的 CGI Perl Script 源代码简单许多。这里需要导入三个包:

```
import Java.io.*;
import Java.Servlet.*;
import Javax.Servlet.*;
```

在 CGI 程序中有很大大一部分用于处理对参数的提取和解码过程,而在 Servlet 程序中,解码传递的参数部分并不需要额外编写。从 Http Servlet class 继承来的基本函数可以自动完成解码的过程。因此,程序员可以在 Servlet service ()方法中直接获取由客户端创建和编码的 Key-value 对。解码后的 Key-value 对可以直接从 Http Servlet class 的 service ()方法的第一个参数中获得。这将极大地减少程序员的劳动量和重复的代码劳动,并降低编码难度。

综上所述,Servlet 在性能、编写难度、可移植性等方面比 CGI 有明显优势。在 WebSphere Application Server 中提供了功能强大的 Servlet API,它们比 JSDK 拥有更多的功能和更优的性能,为 Servlet 的编程提供了很好的支持。随着 WAS 的日益推广和 Java 技术的普及,可以预见,Servlet 技术将取代 CGI,成为对 Web Server 功能扩充的标准技术之一。

## 8.2 Servlet 与 JSP 之间的关系

当 JSP 成为开发动态网站的主要技术时,Servlet 在开发中占据着什么位置呢?网站开发具有两种模式,其中模式二同时用到了 JSP 和 Servlet。在模式二中结合了 JSP 和 Servlet 技术,充分利用了 JSP 和 Servlet 两者的优点。

JSP 技术主要用来表现页面,而 Servlet 技术主要用来完成大量的逻辑处理。也就是说,JSP 主要用来发送给前端的用户,而 Servlet 主要来响应用户的请求,完成请求的逻辑处理。Servlet 充当着控制者的角色,用来负责响应用户的事务处理。

JSP 本身没有任何的业务处理逻辑,它只是简单地检索 Servlet 创建的 JavaBean 或者对象,再将动态的内容插入到预定义的模块中。

Servlet 创建 JSP 需要的 JavaBean 和对象,再根据用户的行为,决定处理哪个 JSP 页面并发送给用户。

由于 Servlet 更适合于后台开发者使用,而且 Servlet 本身需要更多的编程技术,因此 Servlet 本身在页面表现形式上非常得欠缺,远远不如 JSP。

在实际的开发过程中,往往是先把 JSP 页面开发出来,然后再将 JSP 代码转换成 Servlet。这样做的好处是充分利用了 JSP 的页面表现能力,避免了 Servlet 在页面表现方

面得严重不足，大大缩短了开发周期，各尽所能。

例如，当用 Servlet 技术实现如图 8-1 所示的页面时，常常采用的方法是：

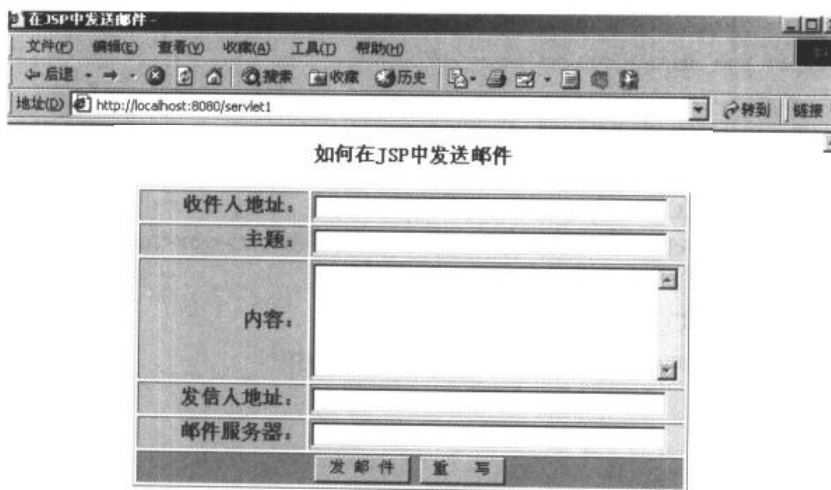


图 8-1 JSP 页面

### 步骤

(1) 首先，用 HTML 编辑工具开发出 JSP 页面，该例子的 JSP 代码如例程 8-3 所示，该代码可以用任何编辑工具进行编写。如可以使用 homesite 编辑工具实现 JSP 代码。

例程 8-3

```
<html>
<head>
<title>在 JSP 中发送邮件</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body bgcolor="#FFFFFF">
<div align="center">
<p><b>如何在 JSP 中发送邮件</b></p>
<form method="post" action="compose.JSP">
<table width="70%" border="1">
<tr>
<td bgcolor="#CCCCFF" width="31%">
<div align="right"><b><font color="#FF0033">收件人地址: </font></b></div>
</td>
<td bgcolor="#FFCCCC" width="69%">
<input type="text" name="recipients" size="40">
</td>
</tr>
<tr>
<td colspan="2">

```

```
<td bgcolor="#CCCCFF" width="31%">
<div align="right"><b><font color="#FF0033">主题: </font></b></div>
</td>
<td bgcolor="#FFCCCC" width="69%">
<input type="text" name="subject" size="40">
</td>
</tr>
<tr>
<td bgcolor="#CCCCFF" width="31%">
<div align="right"><b><font color="#FF0033">内容: </font></b></div>
</td>
<td bgcolor="#FFCCCC" width="69%">
<textarea name="content" cols="40" rows="6"></textarea>
</td>
</tr>
<tr>
<td bgcolor="#CCCCFF" width="31%">
<div align="right"><b><font color="#FF0033">发信人地址: </font></b></div>
</td>
<td bgcolor="#FFCCCC" width="69%">
<input type="text" name="addresser" size="40">
</td>
</tr>
<tr>
<td bgcolor="#CCCCFF" width="31%">
<div align="right"><b><font color="#FF0033">邮件服务器: </font></b></div>
</td>
<td bgcolor="#FFCCCC" width="69%">
<input type="text" name="host" size="40">
</td>
</tr>
<tr bgcolor="#FF9966">
<td colspan="2">
<div align="center">
<input type="submit" name="Submit" value="发 邮 件">
<input type="reset" name="Submit2" value="重 写">
</div>
</td>
</tr>
</table>
</form>
<p>&nbsp;</p>
</div>
</body>
</html>
```

(2) 然后, 将 JSP 的代码移植到 Servlet 相应的位置上。该例子的 Servlet 代码如例程 8-4 所示。

例程 8-4

```
package chapter8;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import sun.net.smtp.*;

public class Servlet1 extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    public void init() throws ServletException {
    }
    /** HTTP Get 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>在 JSP 中发送邮件</title>");
        out.println("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=gb2312\">");
        out.println("</head>");
        out.println("<body bgcolor=\"#FFFFFF\">");
        out.println("<div align=\"center\">");
        out.println("<p><b>如何在 JSP 中发送邮件</b></p>");
        out.println("<form method=\"post\" action=\"compose.JSP\">");
        out.println("<table width=\"70%\" border=\"1\">");
        out.println("<tr> ");
        out.println("<td bgcolor=\"#CCCCFF\" width=\"31%\"> ");
        out.println("<div align=\"right\"><b><font color=\"#FF0033\">收件人地址: </font></b></div>");
        out.println("</td>");
        out.println("<td bgcolor=\"#FFCCCC\" width=\"69%\"> ");
        out.println("<input type=\"text\" name=\"recipients\" size=\"40\">");
        out.println("</td>");
        out.println("</tr>");
        out.println("<tr> ");
        out.println("<td bgcolor=\"#CCCCFF\" width=\"31%\"> ");
        out.println("<div align=\"right\"><b><font color=\"#FF0033\">主题: </font></b></div>");
        out.println("</td>");
        out.println("<td bgcolor=\"#FFCCCC\" width=\"69%\"> ");
        out.println("<input type=\"text\" name=\"subject\" size=\"40\">");
        out.println("</td>");
        out.println("</tr>");
```

```
out.println("<tr> ");
out.println("<td bgcolor=\"#CCCCFF\" width=\"31%\"> ");
out.println("<div align=\"right\"><b><font color=\"#FF0033\">内容: </font></b></div>");
out.println("</td>");
out.println("<td bgcolor=\"#FFCCCC\" width=\"69%\"> ");
out.println("<textarea name=\"content\" cols=\"40\" rows=\"6\"></textarea>");
out.println("</td>");
out.println("</tr>");
out.println("<tr> ");
out.println("<td bgcolor=\"#CCCCFF\" width=\"31%\"> ");
out.println("<div align=\"right\"><b><font color=\"#FF0033\">发信人地址: </font></b></div>");
out.println("</td>");
out.println("<td bgcolor=\"#FFCCCC\" width=\"69%\"> ");
out.println("<input type=\"text\" name=\"addresser\" size=\"40\">");
out.println("</td>");
out.println("</tr>");
out.println("<tr> ");
out.println("<td bgcolor=\"#CCCCFF\" width=\"31%\"> ");
out.println("<div align=\"right\"><b><font color=\"#FF0033\">邮件服务器: </font></b></div>");
out.println("</td>");
out.println("<td bgcolor=\"#FFCCCC\" width=\"69%\"> ");
out.println("<input type=\"text\" name=\"host\" size=\"40\">");
out.println("</td>");
out.println("</tr>");
out.println("<tr bgcolor=\"#FF9966\">");
out.println("<td colspan=\"2\"> ");
out.println("<div align=\"center\"> ");
out.println("<input type=\"submit\" name=\"Submit\" value=\"发 邮 件\">");
out.println("<input type=\"reset\" name=\"Submit2\" value=\"重 写\">");
out.println("</div>");
out.println("</td>");
out.println("</tr>");
out.println("</table>");
out.println("</form>");
out.println("<p>&nbsp;&nbsp;&nbsp;</p>");
out.println("</div>");
out.println("</body>");
out.println("</html>");
}
```

## 8.3 Servlet 应用范围及其缺陷

### 1. Servlet 能够完成以下几种功能

- 创建并返回一个包含基于客户请求性质的动态内容的完整的 HTML 页面。

- 创建可嵌入到现有 HTML 页面中的一部分 HTML 页面 (HTML 片段)。
- 与其他服务器资源 (包括数据库和基于 Java 的应用程序) 进行通信。
- 用多个客户机处理连接, 接收多个客户机的输入, 并将结果广播到多个客户机上。
- 当允许在以单连接方式传送数据的情况下, 在浏览器上打开服务器至 Applet 的新连接, 并将该连接保持在打开状态。在客户机和服务器简单、高效地执行会话的情况下, Applet 也可以启动客户浏览器和服务器之间的连接。可以通过定制协议或标准进行通信。
- 对特殊的处理采用 MIME 类型过滤数据。
- 将定制的处理提供给所有服务器的标准例行程序, Servlet 可以修改如何认证用户。

## 2. 下面是 Servlet 的一些应用范围

- 处理 HTTP 请求

Servlet 能够处理 HTTP 请求, 并且能够传递 HTTP 响应到客户端。

- 用于处理 HTML 表单

通过 HTTP 产生提交数据, 然后 Servlet 可以处理这些数据。

- 允许人们之间的合作

一个 Servlet 能并发处理多个请求, 可以使用同步请求支持系统。

- 转送请求

Servlet 可以转送请求给其他的服务器和 Servlet。这就允许在镜像同样内容的几个服务器之间平衡负载。按照任务类型或组织范围, 可以允许被用来在几个服务器中划分逻辑上的服务器。

- 允许定义激活代理

Servlet 编写者们可以定义彼此之间共同工作的激活代理, 每个代理者是一个 Servlet, 而且代理者能够在他们之间传送数据。

Servlet 的功能及其应用是非常广泛的, 它非常适于服务器端的处理和编程, 并且会长期驻留在它现在的位置。但是, 不能随意地去使用 Servlet, 它本身并不适合每个人。页面设计者可以方便地使用编辑工具开发 JSP 页面, 但是 Servlet 却需要更多的编程技术。也就是说, Servlet 在页面表现形式上缺乏灵活性, 这是 Servlet 的缺陷。

因此, 在实际的开发中, 我们常常利用 JSP 技术的页面表现性、Servlet 技术的逻辑处理能力, 把两者很好地结合在一起使用。

## 8.4 Servlet 的生命周期

### 1. 一般情况下, Servlet 的生命周期可以归纳为以下几点:

- 装载 Servlet

这项操作一般是动态执行的。有些服务器提供了相应的管理功能, 可以在启动的时候就装载 Servlet 并能够初始化特定的 Servlet。

- 创建一个 Servlet 实例



- 调用 Servlet 的 `init()` 方法
- 一个客户端的请求到达服务器
- 创建一个请求对象
- 创建一个响应对象
- 激活 Servlet 的 `Service()` 方法，并传递请求和响应对象
- `service()` 方法获得关于请求对象的信息、处理请求、访问其他资源、获得需要的信息
- `service()` 方法使用响应对象的方法，将响应传回服务器，最终到达客户端。`Service()` 方法可能激活其他方法以便处理请求，如 `doGet()` 或 `doPost()` 或程序员自己开发的方法
- 对于更多的客户端请求，服务器创建新的请求和响应对象，仍然激活此 Servlet 的 `service()` 方法，将这两个对象作为参数传递给它。如此重复以上的循环，但无需再次调用 `init()` 方法。一般情况下，Servlet 只初始化一次
- 当服务器不再需要 Servlet 时或者当服务器关闭的时候，服务器将会调用 Servlet 的 `destroy()` 方法

这就是一个 Servlet 的生命周期。接口规定了在调用 `service()` 方法之前，应该首先完成 Servlet 的 `init()` 方法。同样的，在 Servlet 被销毁之前要先调用 `destroy()` 方法。一旦请求了一个 Servlet，就没有办法阻止 Servlet 容器执行一个完整的生命周期，尽管这不是最优的，但却是接口所要求的。

实际上，Servlet 容器有必要在 Servlet 启动时创建它的一个实例，或者说当 Servlet 首次被调用时，并保持这个 Servlet 实例在内存中，让它对所有的请求进行处理。容器可以决定在任何时期把这个实例从内存中移走。Servlet 有一段时间没有被调用过，或者是容器正在关闭时，容器就会很容易把实例移走。

## 2. Servlet 生命周期的详细描述

### ● 初始化时期

当一个服务器装载 Servlet 时，它运行 Servlet 的 `init()` 方法。

```
public void init(ServletConfig config) throws ServletException
{
    super.init(config);
    //初始化的操作
}
```

需要记住的是要在 `init()` 结束时调用 `super.init()` 方法。`init()` 方法不能反复调用，一旦调用就要重新装载 Servlet。直到服务器调用 `destroy` 方法卸载 Servlet 后才能再调用。

### ● 执行时期

在服务器初始化 Servlet 后，Servlet 就能够用 `service` 方法处理客户端的请求。每个客户端请求有它自己的 `service` 方法，这些方法接收客户端请求，并且发回相应的响应。Servlet 能同时运行多个 `service`。

在 Servlet 执行期间最多的应用是处理客户端的请求并产生一个网页。其代码如下所示：

```
response.setContentType(CONTENT_TYPE);
```

```

PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>在 JSP 中发送邮件</title>");
out.println("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=gb2312\">");
out.println("</head>");
out.println("<body bgcolor=\"#FFFFFF\">");
out.println("</body></html>");
out.close();

```

#### ● Servlet 结束时期

Servlet 一直运行，直到被服务器卸载。在结束的时候收回在 init()方法中使用的资源。在 Servlet 中是通过 destroy()方法来实现的。

```

public void destroy()
{
    //回收在 init()中启用的资源
}

```

综上所述，Servlet 的基本结构应该是这样的：

```

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class Servlet2 extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
        /** 初始化的操作*/
    }
    /**处理 HTTP 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Servlet2</title></head>");
        out.println("<body>");
        out.println("<p>The Servlet has received a GET. This is the reply.</p>");
        out.println("</body></html>");
    }
    /**回收资源*/
    public void destroy() {
    }
}

```

上述 Servlet 的这一基本结构也正说明了 Servlet 的一个生命周期过程。

## 8.5 Servlet 常用类接口

在 ServletsAPI 中, Servlet interface (Servlet 接口) 是非常最重要的。因为所有的 Servlets 都 implement (执行) 这个 interface (接口)。这些 Servlets 或者是直接的执行 Servlet 接口, 或者是通过扩展类(class)执行 Servlet 接口, 如 HttpServlet。这个 Servlet 接口提供了 Servlet 与客户端之间进行通信的联系方法。Servlet 编写者可以在他们开发 Servlet 程序时提供更多的或所有的这种方法。

Servlet 从 Javax 包的 HttpServlet 类扩展, 尽管 Servlet 编写者可以通过直接执行接口来开发 Servlet, 但最通用的开发 Servlet 的方法应该是通过使用 HttpServlet 执行接口来开发 Servlet。

### 8.5.1 HttpServlet

HttpServlet 类是针对使用 HTTP 协议的 Web 服务器的 Servlet 类。HttpServlet 类通过执行 Servlet 接口, 能够提供 HTTP 协议的功能。

在 HttpServlet 类中加入了一些附加的方法, 这些方法可以被 HttpServlet 类中的 Server 方法自动地调用。service 方法是用来协助处理 HTTP 基本请求的 HttpServlet 类中的方法。这些方法有:

#### 1. doGet()用来处理 HTTP 的 GET、头部 HEAD 请求

这个 GET 操作仅仅允许客户从 HTTP 服务器上获取资源。重载此方法的用户自动允许支持方法 HEAD。这个 GET 操作被认为是安全的, 没有任何的负面影响, 对用户来说是很可靠的。打算改变存储数据的请求必须用其他的 HTTP 方法。这要求方法也必须是个安全的操作。方法 doGet 的默认执行将返回一个 HTTP 的 BAD\_Request 错误。

方法 doGet 的格式:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
```

下面是例程 8-5。

例程 8-5

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
```

```
out.println("<html>");
out.println("<body>");
out.println("<head>");
out.println("<title>Hello World!</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Hello World!</h1>");
out.println("</body>");
out.println("</html>");
}
}
```

运行结果如图 8-2 所示。

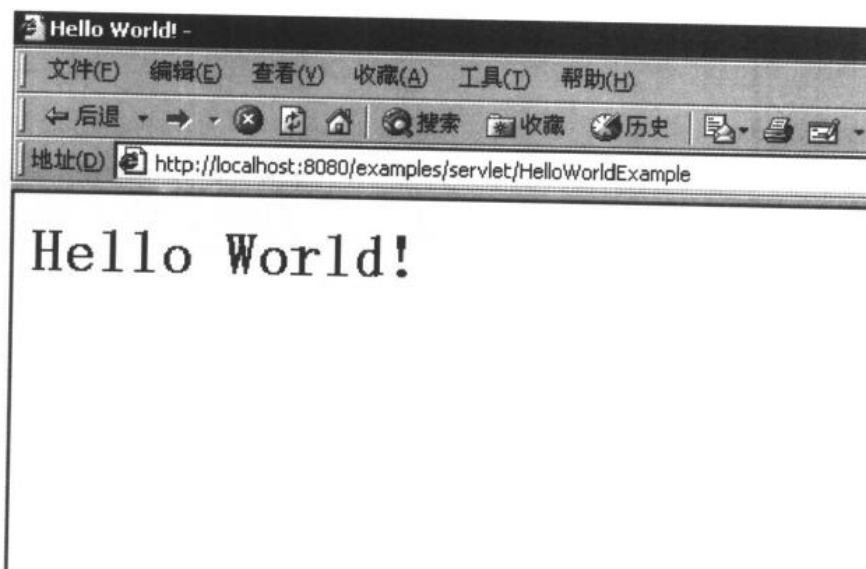


图 8-2 运行结果

## 2. doPost() 用来处理 HTTP 的 POST 请求

这个 POST 操作包含了必须通过 Servlet 执行的请求中的数据。由于它不能立即取得资源，故对于那些涉及到安全性的用户来说，通过 POST 请求操作会有一些副作用。

方法 doPost 的默认执行将返回一个 HTTP 的 BAD\_Request 错误。当编写 Servlet 时，为了支持 POST 操作必须在子类 HttpServlet 中实现（implement）方法。

doPost()方法的格式：

```
public void doPost(HttpServletRequest request, HttpServletResponse res)
    throws IOException, ServletException
```

下面是例程 8-6。

例程 8-6

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
```

```
<title>
Servlet1
</title>
</head>
<body>

<form action="/Servlet1" method="post">
<p>press Submit to post to Servlet Servlet1 </p>
<p>name:::<input type="text" name="name" value=""></p>
<p>password:::<input type="text" name="password" value=""></p>
<p><input type="submit" name="Submit" value="Submit">
<input type="reset" value="Reset"></p>
</form>
</body>
</html>
```

运行结果如图 8-3。

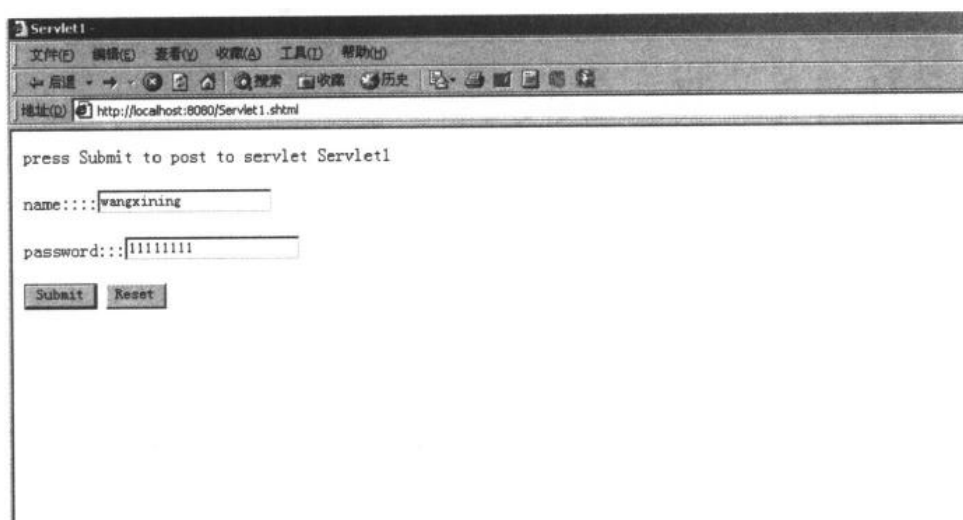


图 8-3 运行结果

Servlet1.java 源代码如例程 8-7 所示。

例程 8-7

```
package sevlet1;
import Javax.Servlet.*;
import Javax.Servlet.http.*;
import Java.io.*;
import Java.util.*;
public class Servlet1 extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
        /** 初始化的操作*/
    }
}
```

```
}  
/**处理 HTTP 请求*/  
public void doPost(HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {  
    response.setContentType(CONTENT_TYPE);  
    PrintWriter out = response.getWriter();  
    out.println("<html>");  
    out.println("<head><title>Servlet1</title></head>");  
    out.println("<body>");  
    out.println("<p>The Servlet has received a POST. This is the reply.</p>");  
    Enumeration e=request.getParameterNames();  
    while(e.hasMoreElements())  
    {  
        String name=(String)e.nextElement();  
        String value=request.getParameter(name);  
        out.println(name+"==="+value);  
    }  
    out.println("</body></html>");  
}  
/**Clean up resources*/  
public void destroy() {  
}  
}
```

运行结果如图 8-4 所示。

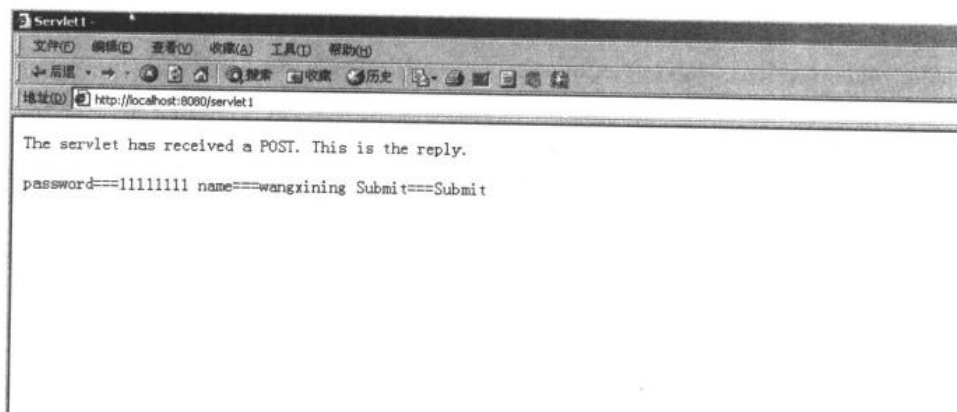


图 8-4 运行结果

上述例子在 Jbuilder5 中很容易调试。

### 3. doPost()用来处理 HTTP 的 POST 请求

此 POST 操作模拟通过 FTP 发送一个文件。对于那些涉及到安全性的用户来说，通过 POST 请求操作也会有一些副作用。

doPost()方法的格式：

```
public void doPost (HttpServletRequest request, HttpServletResponse res)
```



throws IOException, ServletException

#### 4. DoDelete()用来处理 HTTP 的 DELETE 请求

此操作允许客户端请求一个从 server 移出的 URL。对于那些涉及到安全性的用户来说，通过 DELETE 请求操作会有一些副作用。

方法 doDelete 的默认执行将返回一个 HTTP 的 BAD\_Request 错误。当编写 Servlet 时，为了支持 DELETE 操作，必须在子类 HttpServlet 中实现（implement）方法。

doDelete()方法的格式：

```
public void doDelete (HttpServletRequest request, HttpServletResponse res)
throws IOException, ServletException
```

#### 5. DoHead() 用来处理 HTTP 的 HEAD 请求

默认地，它会在无条件的 GET 方法执行时期中运行，但是不返回任何数据到客户端。只返回包含内容信息的长度的 header。由于用到 GET 操作，此方法应该是很安全的（没有副作用）也是可重复使用的。此方法的默认实现（implement）自动地处理了 HTTPDE 的 HEAD 操作并且不需要通过一个子类实现（implement）。

doHead()方法的格式：

```
public void doHead (HttpServletRequest request, HttpServletResponse res)
throws IOException, ServletException
```

#### 6. doOptions()用来处理 HTTP 的 OPTIONS 请求

此操作自动地决定支持什么 HTTP 方法。比如说，如果一个 Servlet 的作者创建 HttpServlet 的子类并重载方法 doGet，然后方法 doOptions 会返回下面的 header：

Allow: GET, HEAD, TRACE, OPTIONS

一般不需要重载方法 doOptions。

doOptions() 方法的格式：

```
public void doOptions (HttpServletRequest request, HttpServletResponse res)
throws IOException, ServletException
```

#### 7. doTrace()用来处理 HTTP 的 TRACE 请求

此方法的默认执行产生一个包含所有在 trace 请求中的 header 的信息的应答（response）。

在开发 Servlet 时，多数情况下需要重载此方法。

doTrace()方法的格式：

```
public void doTrace (HttpServletRequest request, HttpServletResponse res)
throws IOException, ServletException
```

上述几种方法中，在开发以 HTTP 为基础的 Servlet 中，方法 doGet 和方法 doPost 是 Servlet 开发中最常用的方法。

### 8.5.2 HttpServletRequest

HttpServletRequest 被传到 service()方法用来提供客户的请求信息。

HttpServletRequest 接口可以获取由客户端传送的阐述名称，也可以获取客户端正在使用的通信协议，可以获取产生请求并且接收请求的服务器远端主机名和其 IP 地址等一些这样的信息。

HttpServletRequest 接口提供获取数据流的 Servlet、ServletInputStream 方法，这些数据是客户端引用中使用 HTTP POST 和 PUT 方法递交的。一个 ServletRequest 的子类可以让 Servlet 获取更多的协议特性数据。例如：HttpServletRequest 包含获取头部信息的方法有 request.getMethod(), request.getProtocol(), request.getPathInfo()等。

在 Servlet 中，使用 Request 对象获取 HTTP 文件头信息的文件 RequestInfoExample.java 的源代码如例程 8-8 所示。

例程 8-8

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class RequestInfoExample extends HttpServlet {
    ResourceBundle rb = ResourceBundle.getBundle("LocalStrings");
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        String title = rb.getString("requestinfo.title");
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body bgcolor=\"white\">");
        out.println("<a href=\"/examples/Servlets/reqinfo.html\">");
        out.println("<img src=\"/examples/images/code.gif\" height=24 " +
            "width=24 align=right border=0 alt=\"view code\"></a>");
        out.println("<a href=\"/examples/Servlets/index.html\">");
        out.println("<img src=\"/examples/images/return.gif\" height=24 " +
            "width=24 align=right border=0 alt=\"return\"></a>");
        out.println("<h3>" + title + "</h3>");
        out.println("<table border=0><tr><td>");
        out.println(rb.getString("requestinfo.label.method"));
        out.println("</td><td>");
        out.println(request.getMethod());
        out.println("</td></tr><tr><td>");
        out.println(rb.getString("requestinfo.label.requesturi"));
        out.println("</td><td>");
        out.println(request.getRequestURI());
        out.println("</td></tr><tr><td>");
        out.println(rb.getString("requestinfo.label.protocol"));
```

```
out.println("</td><td>");
out.println(request.getProtocol());
out.println("</td></tr><tr><td>");
out.println(rb.getString("requestinfo.label.pathinfo"));
out.println("</td><td>");
out.println(request.getPathInfo());
out.println("</td></tr><tr><td>");
out.println(rb.getString("requestinfo.label.remoteaddr"));
out.println("</td><td>");
out.println(request.getRemoteAddr());
out.println("</table>");
}
public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException
{
    doGet(request, response);
}
}
```

运行结果如图 8-5 所示。



图 8-5 运行结果

### 8.5.3 HttpServletResponse

`HttpServletResponse` 用来向客户端发送响应信息。

`HttpServletResponse` 接口给出相应客户端的 `Servlet` 方法。它允许 `Servlet` 设置内容长度和回应的 `mime` 类型，并且提供输出流 `ServletOutputStream`。

`HttpServletResponse` 子类可以给出更多 `protocol-specific` 容量的信息。例如：

HttpServletResponse 包含允许 Servlet 操作 HTTP-specific 头部信息的方法。

### 1. 定义

```
public interface HttpServletResponse extends ServletResponse
```

描述一个返回到客户端的 HTTP 回应。这个接口允许 Servlet 程序员利用 HTTP 协议规定的头信息。

### 2. 成员变量

```
public static final int SC_CONTINUE = 100;
public static final int SC_SWITCHING_PROTOCOLS = 101;
public static final int SC_OK = 200;
public static final int SC_CREATED = 201;
public static final int SC_ACCEPTED = 202;
public static final int SC_NON_AUTHORITATIVE_INFORMATION = 203;
public static final int SC_NO_CONTENT = 204;
public static final int SC_RESET_CONTENT = 205;
public static final int SC_PARTIAL_CONTENT = 206;
public static final int SC_MULTIPLE_CHOICES = 300;
public static final int SC_MOVED_PERMANENTLY = 301;
public static final int SC_MOVED_TEMPORARILY = 302;
public static final int SC_SEE_OTHER = 303;
public static final int SC_NOT_MODIFIED = 304;
public static final int SC_USE_PROXY = 305;
public static final int SC_BAD_Request = 400;
public static final int SC_UNAUTHORIZED = 401;
public static final int SC_PAYMENT_REQUIRED = 402;
public static final int SC_FORBIDDEN = 403;
public static final int SC_NOT_FOUND = 404;
public static final int SC_METHOD_NOT_ALLOWED = 405;
public static final int SC_NOT_ACCEPTABLE = 406;
public static final int SC_PROXY_AUTHENTICATION_REQUIRED = 407;
public static final int SC_Request_TIMEOUT = 408;
public static final int SC_CONFLICT = 409;
public static final int SC_GONE = 410;
public static final int SC_LENGTH_REQUIRED = 411;
public static final int SC_PRECONDITION_FAILED = 412;
public static final int SC_Request_ENTITY_TOO_LARGE = 413;
public static final int SC_Request_URI_TOO_LONG = 414;
public static final int SC_UNSUPPORTED_MEDIA_TYPE = 415;
public static final int SC_INTERNAL_SERVER_ERROR = 500;
public static final int SC_NOT_IMPLEMENTED = 501;
public static final int SC_BAD_GATEWAY = 502;
public static final int SC_SERVICE_UNAVAILABLE = 503;
public static final int SC_GATEWAY_TIMEOUT = 504;
public static final int SC_HTTP_VERSION_NOT_SUPPORTED = 505;
```

以上 HTTP 产状态码是由 HTTP/1.1 定义的。

### 3. 方法

- **addCookie**

```
public void addCookie(Cookie cookie);
```

在响应中增加一个指定的 `cookie`。可多次调用该方法以定义多个 `cookie`。为了设置适当的头域，该方法应该在响应被提交之前调用。

- **containsHeader**

```
public boolean containsHeader(String name);
```

检查是否设置了指定的响应头。

- **encodeRedirectURL**

```
public String encodeRedirectURL(String url);
```

对 `sendRedirect` 方法使用的指定 URL 进行编码。如果不需要编码，就直接返回这个 URL。之所以提供这个附加的编码方法，是因为在 `redirect` 的情况下，决定是否对 URL 进行编码的规则和一般情况有所不同。所给出的 URL 必须是一个绝对 URL。相对 URL 不能被接收，会抛出一个 `IllegalArgumentException`。

所有提供给 `sendRedirect` 方法的 URL 都应通过这个方法运行，这样才能确保会话跟踪能够在所有浏览器中正常运行。

- **encodeURL**

```
public String encodeURL(String url);
```

对包含 Session ID 的 URL 进行编码。如果不需要编码，就直接返回这个 URL。Servlet 引擎必须提供 URL 编码方法，因为在有些情况下，我们将不得不重写 URL，例如，在响应对应的请求中包含一个有效的 Session，但是这个 Session 不能被非 URL 的（例如 `cookie`）的手段来维持。

所有提供给 Servlet 的 URL 都应通过这个方法运行，这样才能确保会话跟踪能够在所有浏览器中正常运行。

- **sendError**

```
public void sendError(int statusCode) throws IOException;  
public void sendError(int statusCode, String message) throws  
IOException;
```

用给定的状态码发给客户端一个错误响应。如果提供了一个 `message` 参数，这将作为响应体的一部分被发出，否则，服务器会返回错误代码所对应的标准信息。

调用这个方法后，响应立即被提交。在调用这个方法后，Servlet 不会再有更多的输出。

- **sendRedirect**

```
public void sendRedirect(String location) throws IOException;
```

使用给定的路径，给客户端发出一个临时转向的响应（`SC_MOVED_TEMPORARILY`）。给定的路径必须是绝对 URL。相对 URL 将不能被接收，会抛出一个 `IllegalArgumentException`。

这个方法必须在响应被提交之前调用。调用这个方法后，响应立即被提交。在调用这个方法后，Servlet 不会再有更多的输出。

- **setDateHeader**

```
public void setDateHeader(String name, long date);
```



用一个给定的名称和日期值设置响应头，这里的日期值应该是反映自 1970-1-1 日 (GMT) 以来精确到毫秒的长整数。如果响应头已经被设置，新的值将覆盖当前的值。

- **setHeader**

```
public void setHeader(String name, String value);
```

用一个给定的名称和域设置响应头。如果响应头已经被设置，新的值将覆盖当前的值。

- **setIntHeader**

```
public void setIntHeader(String name, int value);
```

用一个给定的名称和整型值设置响应头。如果响应头已经被设置，新的值将覆盖当前的值。

- **setStatus**

```
public void setStatus(int statusCode);
```

这个方法设置了响应的状态码，如果状态码已经被设置，新的值将覆盖当前的值。

以下的几个方法将被取消：

- **encodeRedirectUrl**

```
public String encodeRedirectUrl(String url);
```

该方法被 `encodeRedirectURL` 取代。

- **encodeUrl**

```
public String encodeUrl(String url);
```

该方法被 `encodeURL` 取代。

- **setStatus**

```
public void setStatus(int statusCode, String message);
```

这个方法设置了响应的状态码，如果状态码已经被设置，新的值将覆盖当前的值。如果提供了一个 `message`，它也将被作为响应体的一部分被发送。

#### 8.5.4 HttpSession

这个接口被 Servlet 引擎用来实现在 HTTP 客户端和 HTTP 会话两者的关联。这种关联可能在多处连接和请求中持续一段给定的时间。Session 用来在无状态的 HTTP 协议下越过多个请求页面来维持状态和识别用户。

一个 Session 可以通过 Cookie 或重写 URL 来维持。

方法：

1. **getCreationTime**

```
public long getCreationTime();
```

返回建立 Session 的时间，这个时间表示为自 1970-1-1 日 (GMT) 以来的毫秒数。

2. **getId**

```
public String getId();
```

返回分配给这个 Session 的标识符。一个 HTTP Session 的标识符是一个由服务器来建立和维持的惟一的字符串。

3. **getLastAccessedTime**

```
public long getLastAccessedTime();
```

返回客户端最后一次发出与这个 Session 有关的请求的时间，如果这个 Session 是新建



立的，返回-1。这个时间表示为自 1970-1-1 日（GMT）以来的毫秒数。

#### 4. getMaxInactiveInterval

```
public int getMaxInactiveInterval();
```

返加一个秒数，这个秒数表示客户端在不发出请求时，Session 被 Servlet 引擎维持的最长时间。在这个时间之后，Servlet 引擎可能被 Servlet 引擎终止。如果这个 Session 不会被终止，这个方法返回-1。

当 Session 无效后再调用这个方法会抛出一个 `IllegalStateException`。

#### 5. getValue

```
public Object getValue(String name);
```

返回一个以给定的名字绑定到 Session 上的对象。如果不存在这样的绑定，返回空值。

当 Session 无效后再调用这个方法会抛出一个 `IllegalStateException`。

#### 6. getValueNames

```
public String[] getValueNames();
```

以一个数组返回绑定到 Session 上的所有数据的名称。

当 Session 无效后再调用这个方法会抛出一个 `IllegalStateException`。

#### 7. invalidate

```
public void invalidate();
```

这个方法会终止这个 Session。所有绑定在这个 Session 上的数据都会被清除。并通过 `HttpSessionBindingListener` 接口的 `valueUnbound` 方法发出通告。

#### 8. isNew

```
public boolean isNew();
```

返回一个布尔值以判断这个 Session 是不是新的。如果一个 Session 已经被服务器建立但是还没有收到相应的客户端的请求，这个 Session 将被认为是新的。这意味着，这个客户端还没有加入会话或没有被会话公认。在他发出下一个请求时还不能返回适当的 Session 认证信息。

当 Session 无效后再调用这个方法会抛出一个 `IllegalStateException`。

#### 9. putValue

```
public void putValue(String name, Object value);
```

以给定的名字，绑定给定的对象到 Session 中。已存在的同名的绑定会被重置。这时会调用 `HttpSessionBindingListener` 接口的 `valueBound` 方法。

当 Session 无效后再调用这个方法会抛出一个 `IllegalStateException`。

#### 10. removeValue

```
public void removeValue(String name);
```

取消给定名字的对象在 Session 上的绑定。如果未找到给定名字的绑定的对象，这个方法什么也不做。这时则会调用 `HttpSessionBindingListener` 接口的 `valueUnbound` 方法。

当 Session 无效后再调用这个方法会抛出一个 `IllegalStateException`。

#### 11. setMaxInactiveInterval

```
public int setMaxInactiveInterval(int interval);
```

设置一个秒数，这个秒数表示客户端在不发出请求时，Session 被 Servlet 引擎维持的最长时间。

以下这个方法将被取消。

#### 12. getSessionContext

```
public HttpSessionContext getSessionContext();
```

返回 Session 在其中得以保持的环境变量。这个方法和其他所有 HttpSessionContext 的方法一样被取消了。

### 8.5.5 ServletConfig

在 Servlet 的初始化中，初始化方法使用 ServletConfig 对象作为参数，这个方法应该保存这个对象，以便能够有方法 getServletConfig() 返回。在初始化的方法中，应当保存 ServletConfig 对象，并且重新编写 getServletConfig() 方法，以便它能够从新的位置得到对象。

在下面的例子中，初始化方法就是调用 super.init(config) 方法来管理安排 ServletConfig 对象的，代码如下：

```
public void init(ServletConfig config) throws ServletException
{
    super.init(config);
    //初始化的操作
}
```

### 8.5.6 ServletContext

在服务器上使用 Session 对象来维持同单个客户相关的状态，而当为多个用户的 Web 应用维持一个状态时，则应使用 Servlet 环境 (context)。

ServletContext 对象表示一组 Servlet 共享的资源，在 Servlet API 的 1.0 和 2.0 的版本中，ServletContext 对象仅仅提供了访问有关 Servlet 环境信息的方法。例如：提供了访问服务器名称、MIME 类型映射等方法，和可以将信息写入服务器日志文件的 log() 方法。大部分实现程序都会为一台主机中的所有 Servlet 或每个虚拟机主机提供一个 Servlet 环境。

在 Servlet API 的 2.1 版本中，将 ServletContext 的角色进行了扩展，以便让它承担的责任超出组成一组应用的服务器资源。它允许相同环境中的 Servlet 通过环境 (context) 属性共享信息，其方式与 Session 类似。

ServletContext 既可以用来为一个应用，定义从 URI 到名称的映射，也可以用来让 Servlet 在一个应用中访问共享信息。Servlet 环境的状态信息保存在它的属性中。有三个 ServletContext 方法用于处理环境属性：getAttribute、setAttribute 和 removeAttribute。

## 8.6 JSP 内建对象与 Servlet 中类的对应关系

#### 1. Request 对象

- 在 Servlet 中，使用 Request 对象获取 HTTP 文件头的信息  
RequestHeaderExample.java 的源代码如例程 8-9 所示。

## 例程 8-9

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class RequestHeaderExample extends HttpServlet {

    ResourceBundle rb = ResourceBundle.getBundle("LocalStrings");

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body bgcolor=\"white\">");
        out.println("<head>");

        String title = rb.getString("requestheader.title");
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body>");

        out.println("<a href=\"/examples/Servlets/reqheaders.html\">");
        out.println("<img src=\"/examples/images/code.gif\" height=24 " +
            "width=24 align=right border=0 alt=\"view code\"></a>");
        out.println("<a href=\"/examples/Servlets/index.html\">");
        out.println("<img src=\"/examples/images/return.gif\" height=24 " +
            "width=24 align=right border=0 alt=\"return\"></a>");

        out.println("<h3>" + title + "</h3>");
        out.println("<table border=0>");
        Enumeration e = request.getHeaderNames();
        while (e.hasMoreElements()) {
            String headerName = (String)e.nextElement();
            String headerValue = request.getHeader(headerName);
            out.println("<tr><td bgcolor=\"#CCCCCC\">" + headerName);
            out.println("</td><td>" + headerValue + "</td></tr>");
        }
        out.println("</table>");
    }
}
```

```
public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException
{
    doGet(request, response);
}

}
```

运行结果如图 8-6 所示。

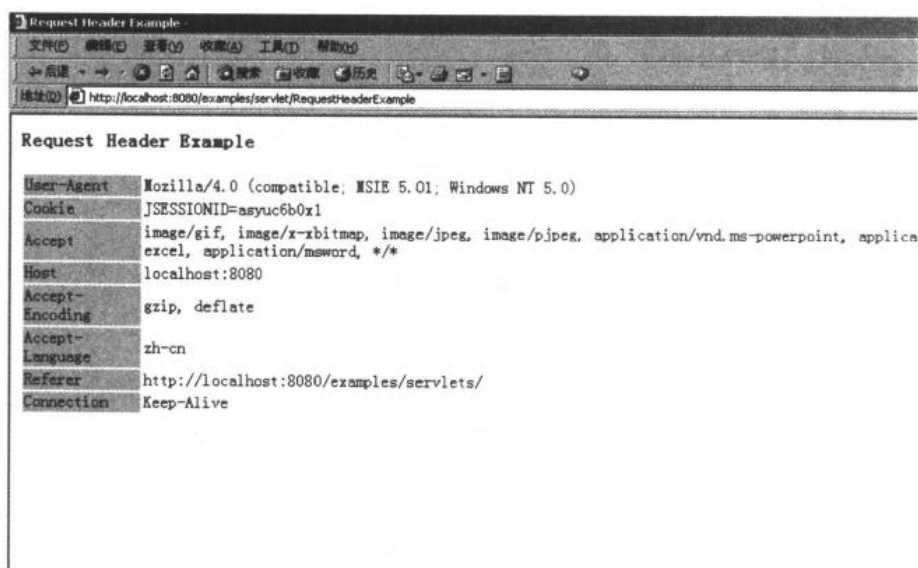


图 8-6 运行结果

- 用 Request 对象获取表单信息

RequestParamExample.java 的源代码如例程 8-10 所示。

例程 8-10

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class RequestParamExample extends HttpServlet {

    ResourceBundle rb = ResourceBundle.getBundle("LocalStrings");

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
```

```
{
response.setContentType("text/html");

PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<body>");
out.println("<head>");

String title = rb.getString("requestparams.title");
out.println("<title>" + title + "</title>");
out.println("</head>");
out.println("<body bgcolor=\"white\">");
out.println("<a href=\"/examples/Servlets/reqparams.html\">");
out.println("<img src=\"/examples/images/code.gif\" height=24 \" +
"width=24 align=right border=0 alt=\"view code\"></a>");
out.println("<a href=\"/examples/Servlets/index.html\">");
out.println("<img src=\"/examples/images/return.gif\" height=24 \" +
"width=24 align=right border=0 alt=\"return\"></a>");

out.println("<h3>" + title + "</h3>");
String firstName = request.getParameter("firstname");
String lastName = request.getParameter("lastname");
out.println(rb.getString("requestparams.params-in-req") + "<br>");
if (firstName != null || lastName != null) {
out.println(rb.getString("requestparams.firstname"));
out.println(" " + firstName + "<br>");
out.println(rb.getString("requestparams.lastname"));
out.println(" " + lastName);
} else {
out.println(rb.getString("requestparams.no-params"));
}
out.println("<P>");
out.print("<form action=\"");
out.print("RequestParamExample\" ");
out.println("method=POST>");
out.println(rb.getString("requestparams.firstname"));
out.println("<input type=text size=20 name=firstname>");
out.println("<br>");
out.println(rb.getString("requestparams.lastname"));
out.println("<input type=text size=20 name=lastname>");
out.println("<br>");
out.println("<input type=submit>");
out.println("</form>");

out.println("</body>");
out.println("</html>");
```

```
}

public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException
{
    doGet(request, response);
}

}
```

运行结果如图 8-7~图 8-8 所示。

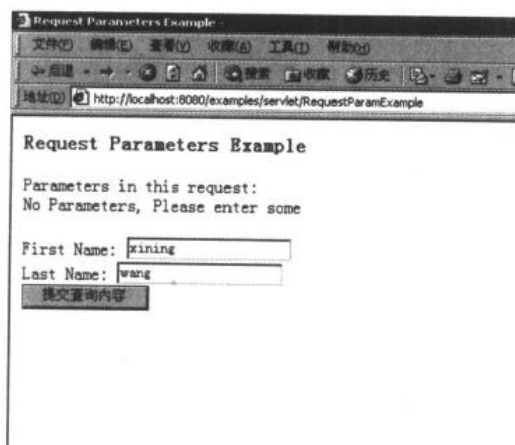


图 8-7 运行结果图 (1)

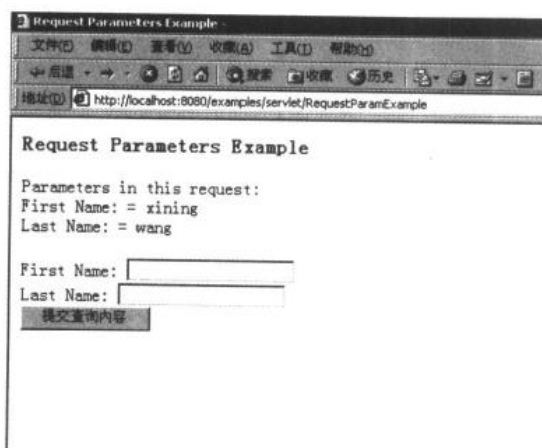


图 8-8 运行结果图 (2)

## 2. Session 对象

在 Servlet 中, 首先用 `request.getSession()` 方法获取 Session 实例, 如 `HttpSession session = request.getSession();` 然后该实例初始化一个 `HttpSession` 对象 (`HttpSession` 类是 `HTTP Session` 类的基类), 然后用 `HTTP Session` 类的方法获取信息, 如例程 8-11 所示。

例程 8-11

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SessionExample extends HttpServlet {

    ResourceBundle rb = ResourceBundle.getBundle("LocalStrings");

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
```



```
{
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<body bgcolor=\"white\">");
    out.println("<head>");

    String title = rb.getString("Sessions.title");
    out.println("<title>" + title + "</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<a href=\"/examples/Servlets/Sessions.html\">");
    out.println("<img src=\"/examples/images/code.gif\" height=24 " +
        "width=24 align=right border=0 alt=\"view code\"></a>");
    out.println("<a href=\"/examples/Servlets/index.html\">");
    out.println("<img src=\"/examples/images/return.gif\" height=24 " +
        "width=24 align=right border=0 alt=\"return\"></a>");

    out.println("<h3>" + title + "</h3>");

    HttpSession Session = request.getSession();
    out.println(rb.getString("Sessions.id") + " " + Session.getId());
    out.println("<br>");
    out.println(rb.getString("Sessions.created") + " ");
    out.println(new Date(Session.getCreationTime()) + "<br>");
    out.println(rb.getString("Sessions.lastaccessed") + " ");
    out.println(new Date(Session.getLastAccessedTime()));

    String dataName = request.getParameter("dataname");
    String dataValue = request.getParameter("datavalue");
    if (dataName != null && dataValue != null) {
        Session.setAttribute(dataName, dataValue);
    }

    out.println("<P>");
    out.println(rb.getString("Sessions.data") + "<br>");
    Enumeration names = Session.getAttributeNames();
    while (names.hasMoreElements()) {
        String name = (String) names.nextElement();
        String value = Session.getAttribute(name).toString();
        out.println(name + " = " + value + "<br>");
    }

    out.println("<P>");
    out.print("<form action=\"\">");
```

```
        out.print(response.encodeURL("SessionExample"));
    out.print("\n ");
    out.println("method=POST>");
    out.println(rb.getString("Sessions.dataname"));
    out.println("<input type=text size=20 name=dataname>");
    out.println("<br>");
    out.println(rb.getString("Sessions.datavalue"));
    out.println("<input type=text size=20 name=datavalue>");
    out.println("<br>");
    out.println("<input type=submit>");
    out.println("</form>");

    out.println("<P>GET based form:<br>");
    out.print("<form action=\"");
        out.print(response.encodeURL("SessionExample"));
    out.print("\n ");
    out.println("method=GET>");
    out.println(rb.getString("Sessions.dataname"));
    out.println("<input type=text size=20 name=dataname>");
    out.println("<br>");
    out.println(rb.getString("Sessions.datavalue"));
    out.println("<input type=text size=20 name=datavalue>");
    out.println("<br>");
    out.println("<input type=submit>");
    out.println("</form>");

    out.print("<p><a href=\"");
        out.print(response.encodeURL("SessionExample?dataname=foo&datavalue=bar"));
        out.println("\n ">URL encoded </a>");

    out.println("</body>");
    out.println("</html>");

    out.println("</body>");
    out.println("</html>");
}

public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException
{
    doGet(request, response);
}

}
```

运行结果如图 8-9~图 8-10 所示。

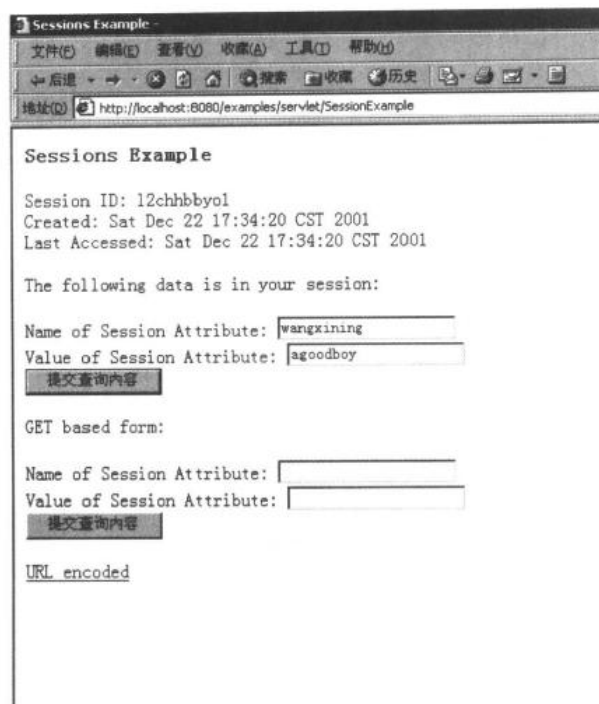


图 8-9 运行结果图 (1)

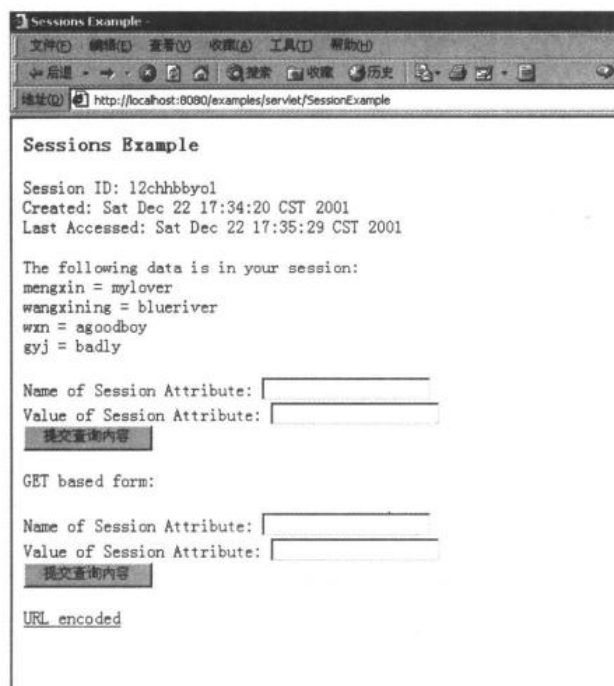


图 8-10 运行结果图 (2)

### 3. Cookie 对象

在 Servlet 中，直接定义一个 Cookie 数组，然后用 `request.getCookies()` 方法去初始化。

如：`Cookie[] cookies = request.getCookies();`

CookieExample.java 的运行结果如图 8-11 所示，其源代码如例程 8-12 所示。

例程 8-12

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class CookieExample extends HttpServlet {
    ResourceBundle rb = ResourceBundle.getBundle("LocalStrings");

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body bgcolor=\"white\">");
        out.println("<head>");
        String title = rb.getString("cookies.title");
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<a href=\"/examples/Servlets/cookies.html\">");
        out.println("<img src=\"/examples/images/code.gif\" height=24 " +
            "<width=24 align=right border=0 alt=\"view code\"></a>");
        out.println("<a href=\"/examples/Servlets/index.html\">");
        out.println("<img src=\"/examples/images/return.gif\" height=24 " +
            "<width=24 align=right border=0 alt=\"return\"></a>");
        out.println("<h3>" + title + "</h3>");
        Cookie[] cookies = request.getCookies();
        if (cookies.length > 0) {
            out.println(rb.getString("cookies.cookies") + "<br>");
            for (int i = 0; i < cookies.length; i++) {
                Cookie cookie = cookies[i];
                out.print("Cookie Name: " + cookie.getName() + "<br>");
                out.println("  Cookie Value: " + cookie.getValue() +
                    "<br><br>");
            }
        } else {
            out.println(rb.getString("cookies.no-cookies"));
        }
        String cookieName = request.getParameter("cookieName");
        String cookieValue = request.getParameter("cookieValue");
        if (cookieName != null && cookieValue != null) {
            Cookie cookie = new Cookie(cookieName, cookieValue);
```

```
response.addCookie(cookie);
out.println("<P>");
out.println(rb.getString("cookies.set") + "<br>");
out.print(rb.getString("cookies.name") + " " + cookieName +
    "<br>");
out.print(rb.getString("cookies.value") + " " + cookieValue);
}
out.println("<P>");
out.println(rb.getString("cookies.make-cookie") + "<br>");
out.print("<form action=\"");
out.println("CookieExample\" method=POST>");
out.print(rb.getString("cookies.name") + " ");
out.println("<input type=text length=20 name=cookieName><br>");
out.print(rb.getString("cookies.value") + " ");
out.println("<input type=text length=20 name=cookieValue><br>");
out.println("<input type=submit></form>");
out.println("</body>");
out.println("</html>");
}
public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException
{
    doGet(request, response);
}
}
```



图 8-11 运行结果图

# 第9章 JSP 与 Servlet 结合编程

在本章中，讲述了 Servlet 技术与 JSP 技术是如何结合使用的。首先讲述了 JSP 技术网站开发的两种模式，接着讲述了这两种模式的实例。9.3 和 9.4 节则讲述了在 JSP 中是如何发送邮件的。

通过对本章的学习，将会掌握 JSP 开发中的两种工作模式及其具体的应用。

## 9.1 JSP 技术网站开发的两种模式

下面我们来分别讲述网站开发的两种模式。

### 9.1.1 模式一：JSP+JavaBeans

目前，JSP 技术已经成为一种受大多数大型企业或中小型企业喜爱的动态网站开发技术。越来越多的技术人员也逐步成为 JSP 技术的推崇者。

JSP 技术正是利用了 Java 的“一次开发，处处使用”的性能，成为了网站开发技术人员的首选技术。当然，JSP 技术的最大优势在于它能够将页面的表现形式和页面的商业逻辑分开。

JSP 网站开发技术标准给出了两种使用 JSP 的方式。这些方式，都可以归纳为模式一和模式二。

模式一，就是指 JSP+JavaBeans 技术。在模式一中，JSP 页面独自响应请求并将处理结果返回客户。所有的数据通过 Bean 来处理，JSP 实现页面的表现。模式一技术也实现了页面的表现——和页面的商业逻辑相分离。

大量使用模式形式，常常会导致页面被嵌入大量的脚本语言或者 Java 代码段。当需要处理的商业逻辑很复杂时，这种情况会变得非常得糟糕。大量的内嵌代码使得整个页面程序变得异常复杂。对于前端界面设计人员来说，这简直是不可思议的事情。

这种情况在大型项目中常见，这样也造成了代码的开发和维护将出现困难，造成了不必要的资源浪费。在任何项目中，这样的模式多少总会导致定义不清的响应和项目管理的困难。

综上所述，模式一不能够满足大型应用的要求，尤其是大型的项目。但是，模式一可以很好地满足小型应用的需要，在简单的应用中，可以考虑使用模式一。

### 9.1.2 模式二：JSP+Servlet+JavaBeans

Servlets 技术是一种采用 Java 技术来实现 CGI 功能的一种技术。Servlets 技术是运行



在 Web 服务器上，用来生成 Web 页面。Servlets 技术非常适于服务器端的处理和编程，并且 Servlets 会长期驻留在他们现在的位置。

但是，在实际的项目开发过程中，页面设计者可以方便地使用普通的 HTML 工具来开发 JSP 页面，Servlets 却更适合于后端开发者使用，开发 Servlets 需要的工具是 Java 集成开发环境。也就是说，Servlets 技术更需要技术人员更多的编程。

模式二，就是指 JSP+Servlet+JavaBeans 技术。在模式二中，结合了 JSP 和 Servlet 技术，模式二充分利用了 JSP 和 Servlet 两种技术原有的优点。

在模式二中，通过 JSP 技术来表现页面，通过 Servlet 技术来完成大量的事务处理工作。在模式二中，Servlet 用来处理请求的事务，充当着一个控制者的角色，并负责向客户发送请求。Servlet 创建 JSP 需要的 Bean 和对象，然后根据用户的请求行为，决定将哪个 JSP 页面发送给用户。



JSP 页面中没有任何商业处理逻辑，所有的商业处理逻辑均出现在 Servlet 中。JSP 页面只是简单地检索 Servlet 先前创建的 Bean 或者对象，再将动态内容插入到预定义的模版中。

从开发的观点看，模式二具有更清晰的页面表现，清楚的开发者角色划分，可以充分地利用开发小组中的界面设计人员。这些优势在大型项目开发中表现得尤为突出，使用这一模式，可以充分发挥每个开发者各自的特长，界面设计开发人员可以充分发挥自己的设计才能，来体现页面的表现形式，程序编写人员则可以充分发挥自己的商务处理逻辑思维，来实现项目中的业务处理。

在目前大型项目开发中，模式二更被采用。

在后面的章节中，我们将通过实例来讲述一下这两种模式的不同。

## 9.2 两种模式的实例

下面，我们以 WebMail 为例来讲述上述两种模式的工作原理。我们用两种不同的模式分别实现了在 Web 页面中发送信息的功能，也就是常用的 WebMail。

由于在下面的例子中，我们用到了数据库、SQL 语句以及 JDBC 技术，所以建议读者在阅读后面的有关章节之后，再来阅读下面的实例。

例子中，我们用到的数据库是 Microsoft SQL Server 2000。整个建立数据库的步骤大致如下：



(1) 打开“Enterprise Manager (企业管理器)”。

(2) 在“数据库”一栏中，单击右键，在图 9-1 所示界面的快捷菜单中选择【新建数据库】，将出现如图 9-2 所示的界面。

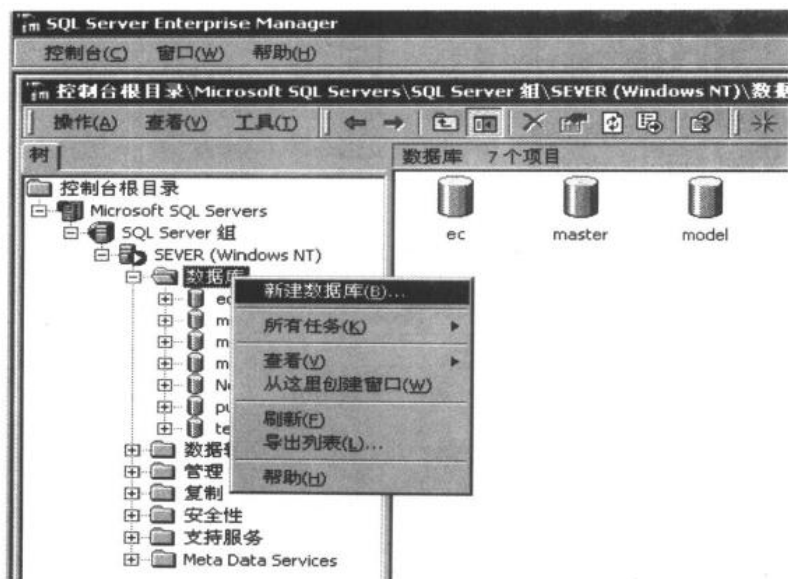


图 9-1 企业管理器界面

(3) 在图 9-2 中输入数据库的名称 **WebMail**，其他设置默认，然后单击【确定】按钮。这时，我们就已经成功地建立了一个新数据库，其名称是 **WebMail**。

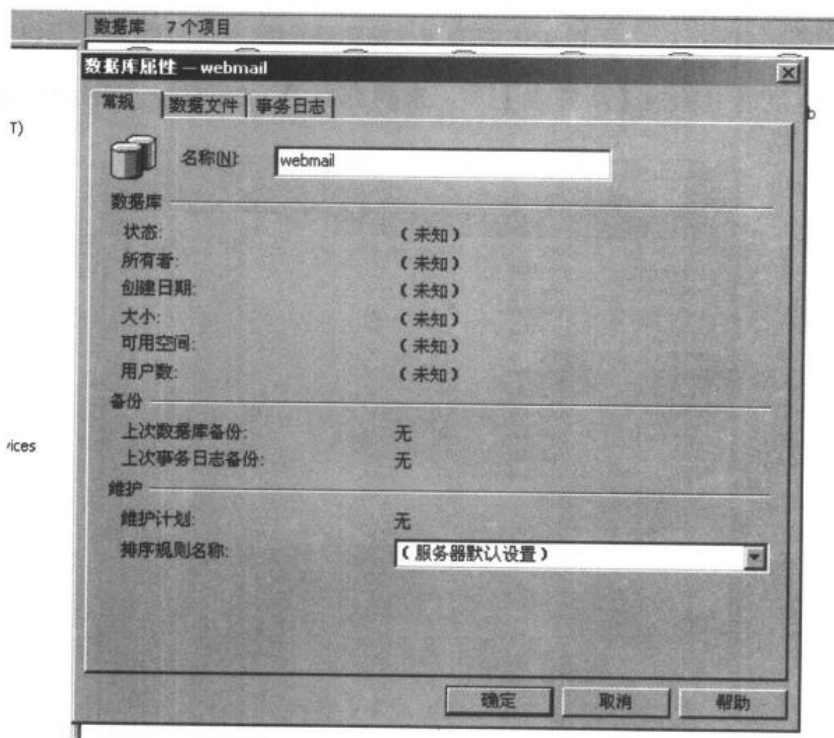


图 9-2 新建数据库界面

(4) 然后单击“**WebMail**”字样前的加号，在“表”字样的位置点击鼠标右键，将出现如图 9-3 所示的界面。

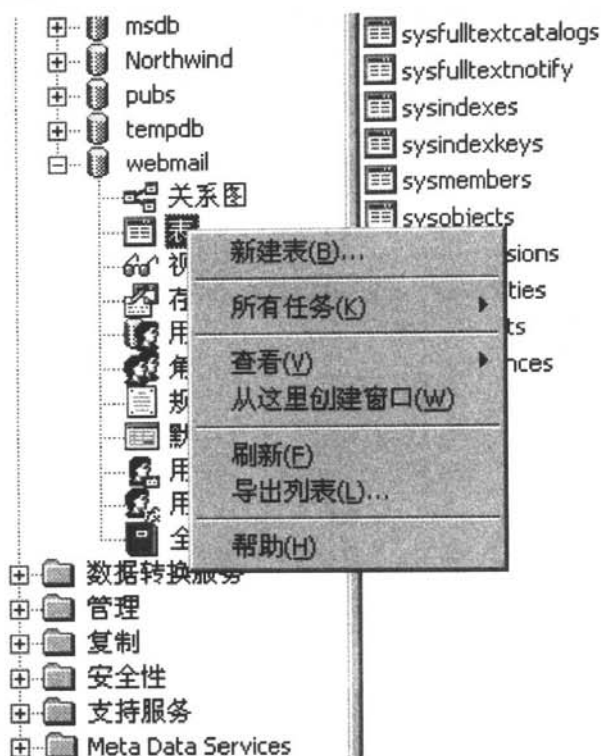


图 9-3 新建表快捷菜单界面

(5) 从快捷菜单中单击【新建表】项，来建立新表 member，表的结构如图 9-4 所示。

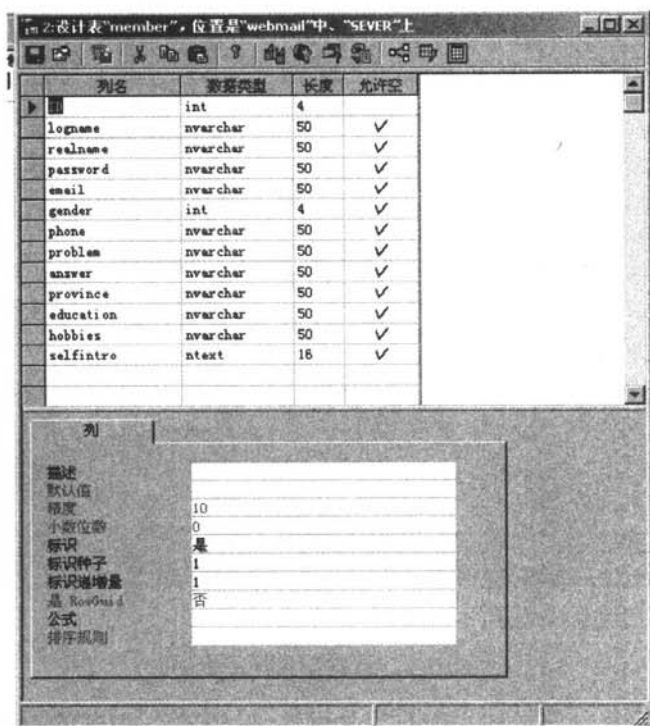


图 9-4 新建 member 结构图

(6) 按照同样的方法，我们又新建了表 Inbox，表的结构如图 9-5 所示。

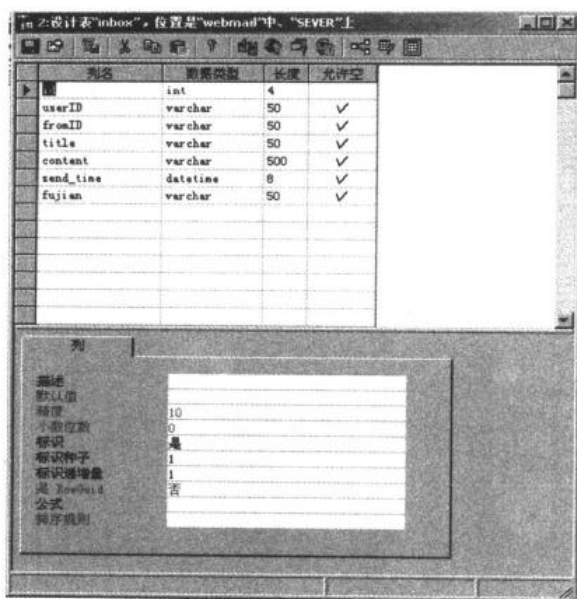


图 9-5 表 Inbox 结构图

至此，我们的数据库已经全部建立。当然，我们的这个实例在功能上肯定不全面。譬如：在实际中，我们应该需要建立存储垃圾信件的表，也需要存放草稿信件的表。至于这些功能，希望读者自己练习一下。

下面，我们按照如下步骤，来建立一个新的系统数据源。



(1) 打开控制面板中的管理工具，如图 9-6 所示。

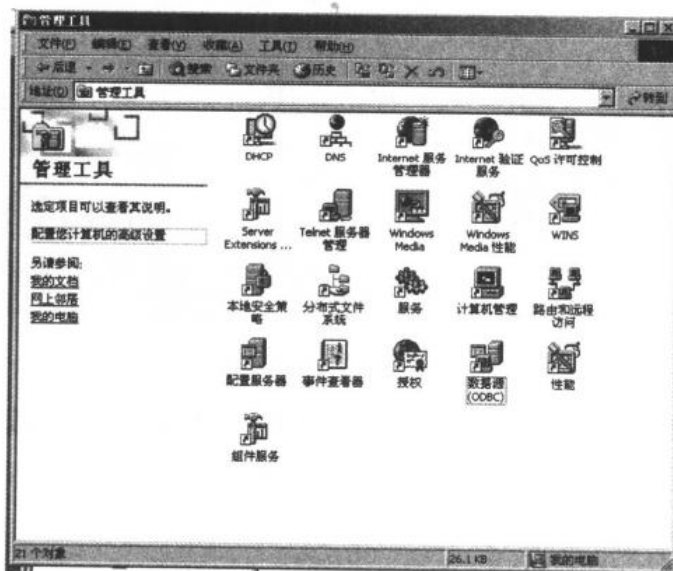


图 9-6 管理工具界面



(2) 单击“数据源 (ODBC)”图标, 将出现如图 9-7 所示的界面。

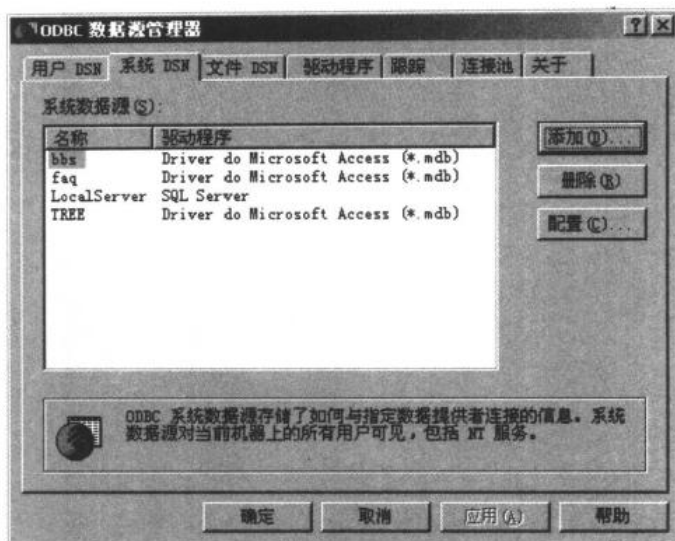


图 9-7 数据源管理器界面

(3) 在图单击“系统 DSN” (系统数据源) 标签。

(4) 然后再单击【添加】按钮, 将出现如图 9-8 所示的界面。



图 9-8 创建新数据源界面

(5) 在对话框中, 选择 SQL Server 驱动程序, 然后单击【完成】按钮, 将出现“建立新的数据源到 SQL Server”对话框, 如图 9-9 所示。

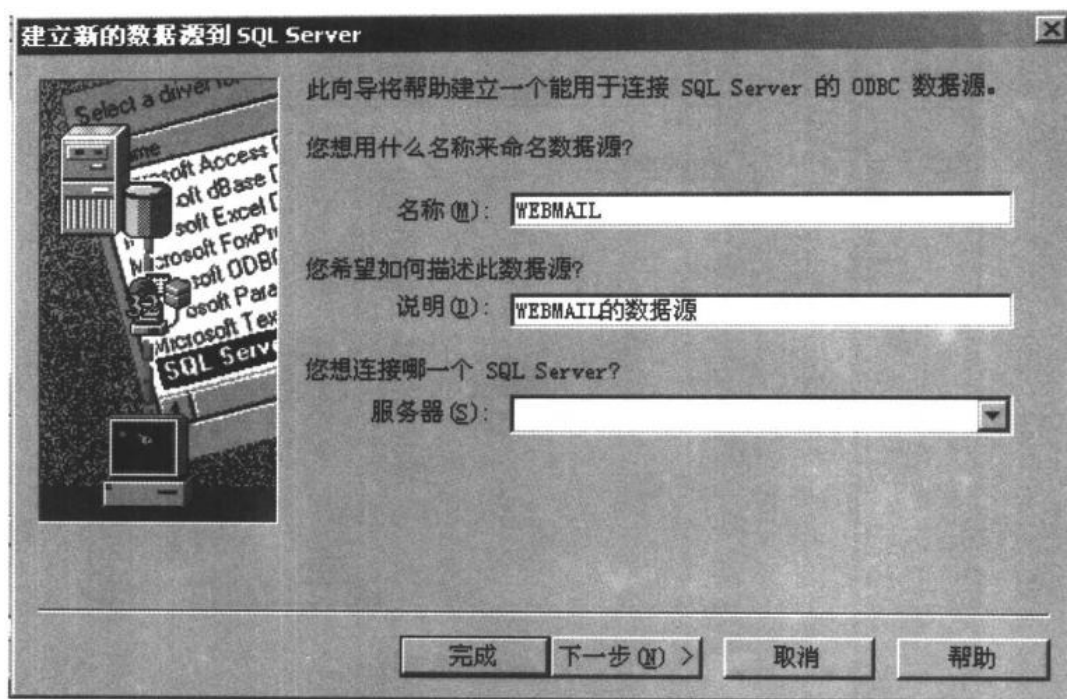


图 9-9 建立新的数据源到 SQL Server 界面

(6) 输入要新建的数据源的名字, 如 WebMAIL。在描述数据源的说明文本框中, 输入一个描述语, 如 WebMAIL 的数据源。在服务器文本框中, 输入安装了 SQL Server 的服务器名称。如果 SQL Server 与 Web 服务器是安装在同一台机器上, 则可以输入 Local 或者不输入。然后单击【下一步】按钮, 将出现如图 9-10 所示的界面。

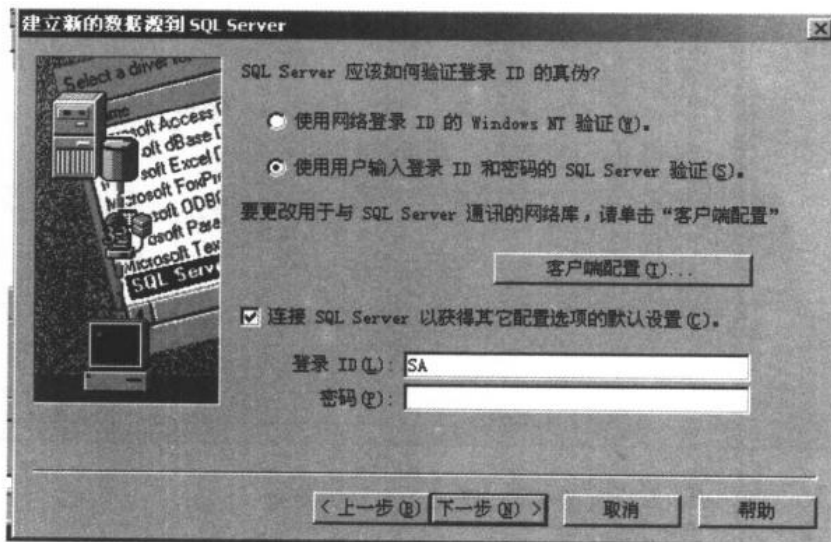


图 9-10 验证登录 ID 界面

对认证方式进行设定, 在本例子中, 我们可以以 SA 登录, 密码为空。在实际的应用中, 要设置其他用户或添加密码, 以便能保障系统的安全性。

(7) 输入完后, 单击【下一步】按钮, 将出现如图 9-11 所示的界面。



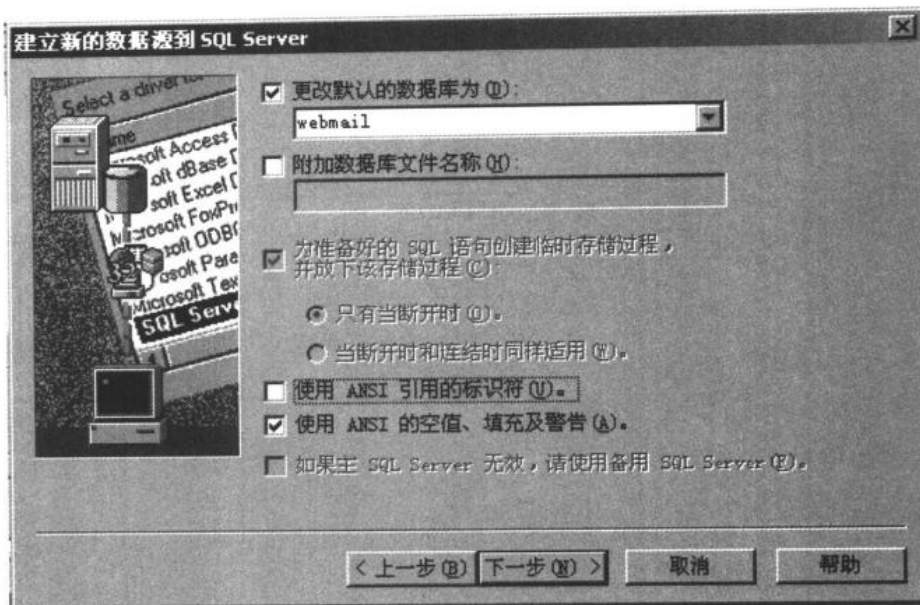


图 9-11 指定数据库界面

指定一个默认的数据库，如上图所示。选择表所在的数据库作为默认数据库。在本例中，我们选择 WebMAIL 数据库作为默认数据库。

(8) 然后单击【下一步】按钮，将出现如图 9-12 所示的界面。

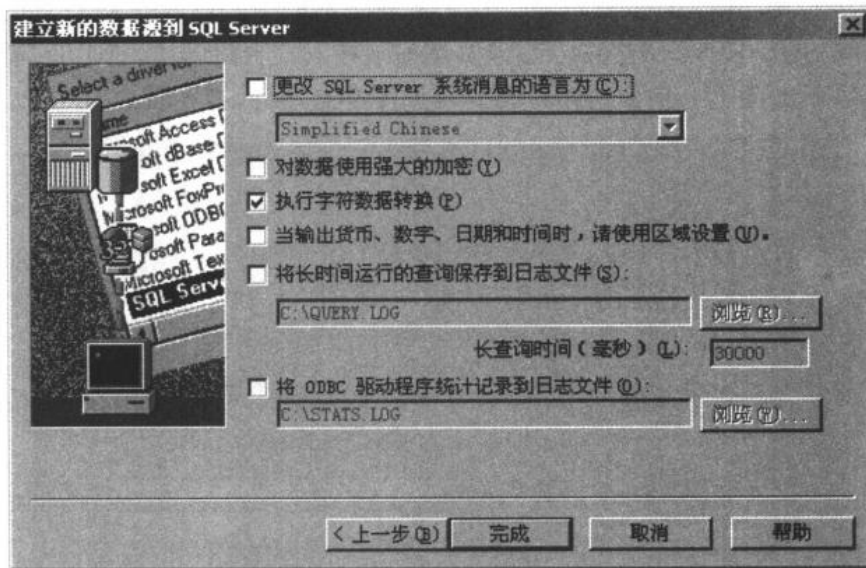


图 9-12 更改语言种类界面

在这个界面中，您可以更改 SQL Server 系统信息的语言种类，也可以将长时间运行的查询保存到您指定的日志文件，也可以将 ODBC 驱动程序统计纪录到日志文件等。如果不做任何改动，则会采用默认设置。

(9) 设置完后，单击【完成】按钮，将出现如图 9-13 所示的界面。

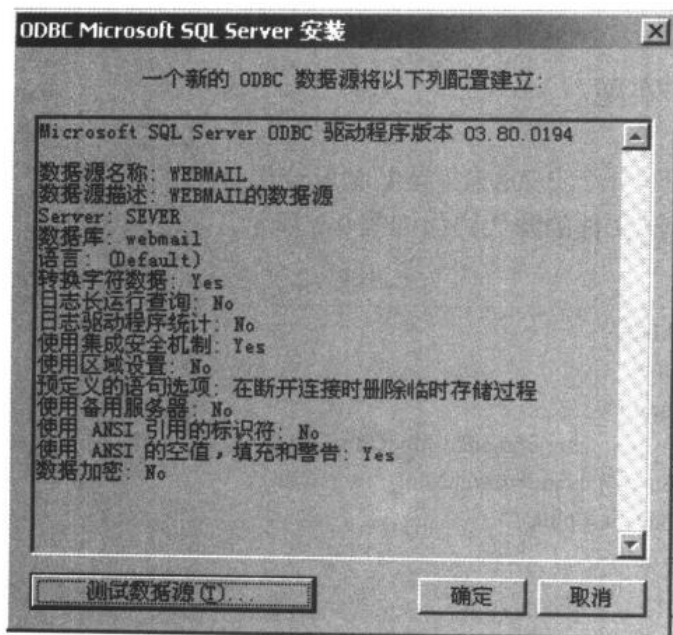


图 9-13 测试数据源界面

(10) 在这个界面中, 提供了一个测试数据源的机会。单击【测试数据源】按钮, 进行数据源测试, 将出现如图 9-14 所示的界面。

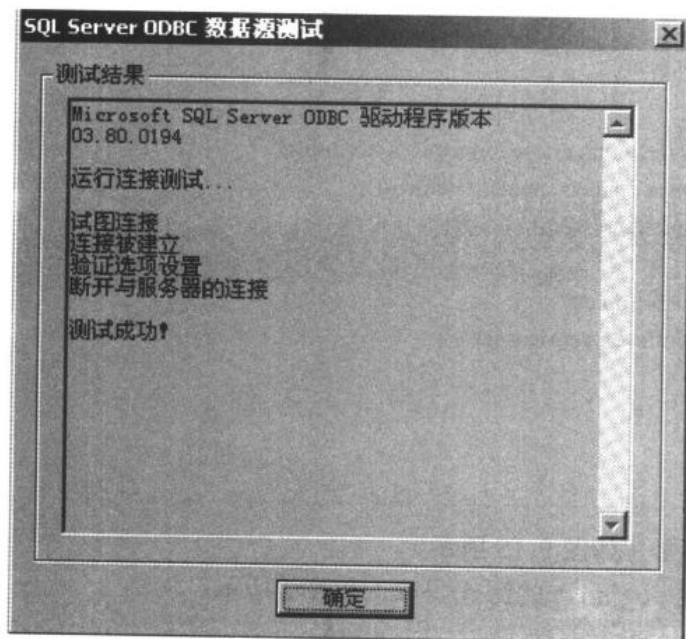


图 9-14 数据源测试结果界面

(11) 出现如上图所示的界面, 说明数据源测试成功, 单击【确定】按钮, 添加了这个新的数据源。

现在我们已经建立了一个名为 WebMAIL 的新数据源。可以使用这个数据源连接到

Microsoft SQL Server。

### 9.2.1 使用模式一实现

在本例中，用到了一个 Bean 来封装数据库连接。  
Webmail.conn.java 文件的源代码如例程 9-1 所示。

例程 9-1

```
package WebMail;
import java.sql.*;
public class conn {
    String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
    String sConnStr = "jdbc:odbc:WebMail";
    Connection connect = null;
    ResultSet rs = null;
    public conn() {
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex)
            System.err.println(ex.getMessage());
        }
        return rs;
    }
}
```

#### 1. 用户登录界面（如图 9-15 所示）

用户登录的文件 login.jsp 的源代码如例程 9-2 所示。

例程 9-2

```
<html>
<head>
    <title>Untitled</title>
</head>
```

```

<body>

<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">

    </td>
    <td width="10" height="100%"></td>
    <td width="1" height="100%" bgcolor="#3399ff"></td>
    <td width="10" height="100%"></td>
    <td width="589" height="331" valign="top" background="images/bg1.gif">
      <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
        <tr>
          <td width="100%" colspan="2" height="20" bgcolor="#3399ff">&nbsp;<font
color="#ffffff">成员登录</font>
          </td>
        </tr>
        <tr>
          <td colspan="2">
            <form action="login_ok.jsp" method="post">
              <tr><td align="right" height="32" width="40%">名字: </td>
              <td>
                <input type="hidden" name="returl" value="">
                <input type="text" name="logname" value="">
              </td>
            </tr>
            <tr>
              <td align="right" height="32">密码: </td>
              <td>
                <input type="password" name="logpass">
              </td>
            </tr>
            <tr>
              <td align="center" colspan="2" height="32">|
                <a href="findpass.jsp">密码忘了, 找一下</a> |
                <a href="reg.jsp">注册新用户</a> |
                <input type="submit" name="login" value="登录!">
              </td>
            </tr>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
</body>

```

&lt;/html&gt;

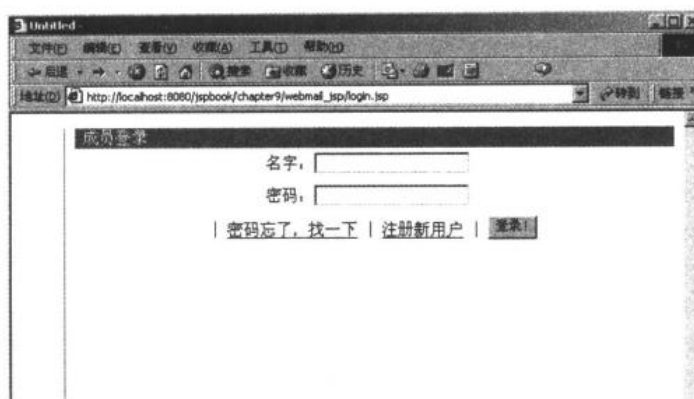


图 9-15 用户登录界面

如果用户成功登录, 则进入 main.jsp 界面; 如果用户不能成功登录, 则会让用户再次登录, 则进入 Login\_ok.jsp 界面。

源代码如例程 9-3 所示。

例程 9-3

```
<%@ page contentType="text/html;charset=gb239" %>
<%@page language="java" import="java.sql.*"%>
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>

<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>

<%!
String logname,logpass;
boolean loginAttempt = false;
boolean loginOK = false;
String errorMessage = "请您登录";
```



```
%>
<%
if(request.getParameterValues("login") != null
&&request.getParameterValues("logname") != null
&&request.getParameterValues("logpass") != null)
{
    loginAttempt = true;
}
if (loginAttempt)
{
    logname=request.getParameter("logname");
    logpass=request.getParameter("logpass");
    logname=getStr(logname);
    logpass=getStr(logpass);
    String sql="select * from member where logname='"+logname+"' and password='"+logpass+"'";
    //out.println(sql);
    ResultSet RS=userBean.executeQuery(sql);
    int rowcount=0;
    try
    {
        while(RS.next())
        {
            rowcount++;
        }
    }
    catch(Exception e)
    {

    }
    //count.....
    // out.println(rowcount);
    if(rowcount!=0)
    {
        errorMessage="成功登录";
        session.setAttribute("username",logname);
        loginOK=true;

        if(loginOK){
            response.sendRedirect("main.jsp");
        }
    }else{
        errorMessage="您的用户名或者密码不正确";
        session.setAttribute("username","");
    }
}
```



```
}
%>
```

```
<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">

      </td>
      <td width="10" height="100%"></td>
      <td width="1" height="100%" bgcolor="#3399ff"></td>
      <td width="10" height="100%"></td>
      <td width="589" height="331" valign="top" background="images/bg1.gif">
        <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
          <tr>
            <td width="100%" colspan="2" height="20" bgcolor="#3399ff">&nbsp;<font
color="#ffffff">成员登录</font>
            </td>
          </tr>
          <tr>
            <td align="right" height="32" width="40%">
              <font color=red><%=errorMessage%></font>
            </td>
          </tr>
          <tr>
            <td align="right" colspan="2" height="32" colspan="2">
              <form action="" method="post">
                <tr>
                  <td align="right" height="32" width="40%">名字: </td>
                  <td>
                    <input type="hidden" name="returl" value="">
                    <input type="text" name="logname" value="">
                  </td>
                </tr>
                <tr>
                  <td align="right" height="32" width="40%">密码: </td>
                  <td>
                    <input type="password" name="logpass">
                  </td>
                </tr>
                <tr>
                  <td align="center" colspan="2" height="32">|
                    <a href="findpass.jsp">密码忘了, 找一下</a> |
                    <a href="reg.jsp">注册新用户</a> |
                    <input type="submit" name="login" value="GO!">
                  </td>
                </tr>
              </form>
            </td>
          </tr>
          <tr>
            <td colspan="2" align="right">&nbsp;</td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
```

```

        </tr>

    </table>

    </td>
</tr>
</table>
</body>
</html>

```

## 2. 用户注册界面（如图 9-16 所示）

reg.jsp 文件的源代码如例程 9-4 所示。

例程 9-4

```

<html>
<head>
    <title>Untitled</title>
</head>

<body>
    <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
        <tr align=center>
            <td width="100%" colspan="4" height="20" bgcolor="#3399ff">&nbsp;<font
color="#ffffff">新成员注册新成员注册新成员注册新成员注册</font>
            </td>
        </tr>

        <tr>
            <td width="100%" align="center" colspan="4" height="36"> 粗体内容必须填写 </td>
        </tr>
        <form name="form1" method="post" action="reg_ok.jsp">
            <tr>
                <td align="right"><b>登录名称: </b></td>
                <td>&nbsp;</td>
                <td colspan="2">
                    <input type="text" name="logname" value="wangxining">
                </td>
            </tr>
            <tr>
                <td align="right"><b>真实姓名: </b></td>
                <td>&nbsp;</td>
                <td colspan="2">
                    <input type="text" name="realname" value="wangwang">
                </td>
            </tr>
        </form>
    </table>

```

```
<tr>  
    <td align="right"><b>您的密码:</b></td>  
    <td>&nbsp;</td>  
    <td colspan="2">  
        <input type="password" name="passwd1" value="blueriver">  
    </td>  
</tr>  
<tr>  
    <td align="right"><b>密码确认:</b></td>  
    <td>&nbsp;</td>  
    <td colspan="2">  
        <input type="password" name="passwd2" value="blueriver">  
    </td>  
</tr>  
<tr>  
    <td align="right"><b>电子邮件:</b></td>  
    <td>&nbsp;</td>  
    <td colspan="2">  
        <input type="text" name="email" value="wangxining@263.net">  
    </td>  
</tr>  
<tr>  
    <td align="right"><b>性&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~</td>  
    <td>&nbsp;</td>  
    <td colspan="2">  
        <input type="radio" name="Gender" value="0" checked>  
        男  
        <input type="radio" name="Gender" value="1">  
        女 </td>  
</tr>  
<tr>  
    <td align="right">联系电话:</td>  
    <td>&nbsp;</td>  
    <td colspan="2">  
        <input type="text" name="phone" value="">  
    </td>  
</tr>  
<tr>  
    <td align="right">密码提示问题:</td>  
    <td>&nbsp;</td>  
    <td colspan="2">  
        <input type="text" name="problem" value="">  
    </td>  
</tr>  
<tr>  
    <td align="right">提示答案:</td>
```

```
<td>&nbsp;</td>
<td colspan="2">
  <input type="text" name="answer" value="">
</td>
</tr>
<tr>
<td align="right">省市: </td>
<td>&nbsp;</td>
<td colspan="2">
  <select name="Province" size="1">
    <option value="null"      >请选择</option>
    <option value="Beijing"   >北京</option>
    <option value="Tianjin"    >天津</option>
    <option value="Shanghai"   >上海</option>
    <option value="Chongqing"  >重庆</option>
    <option value="Hebei"      >河北</option>
    <option value="Henan"      >河南</option>
    <option value="Heilongjiang">黑龙江</option>
    <option value="Jinlin"     >吉林</option>
    <option value="Shenyang"    >辽宁</option>
    <option value="Neimenggu"   >内蒙古</option>
    <option value="Hainan"      >海南</option>
    <option value="Shanxi"      >山西</option>
    <option value="Shanxi3"     >陕西</option>
    <option value="Shandong"    >山东</option>
    <option value="Jiangsu"     >江苏</option>
    <option value="Zhejiang"    >浙江</option>
    <option value="Anhui"       >安徽</option>
    <option value="Jiangxi"     >江西</option>
    <option value="Fujian"      >福建</option>
    <option value="Gansu"       >甘肃</option>
    <option value="Ningxia"     >宁夏</option>
    <option value="Qinghai"     >青海</option>
    <option value="Xinjiang"    >新疆</option>
    <option value="Hubei"       >湖北</option>
    <option value="Hunan"       >湖南</option>
    <option value="Guangxi"     >广西</option>
    <option value="Sichuan"     >四川</option>
    <option value="Guizhou"     >贵州</option>
    <option value="Yunnan"      >云南</option>
    <option value="Xizang"      >西藏</option>
    <option value="Hongkong"    >香港</option>
    <option value="Aomen"       >澳门</option>
    <option value="Taiwan"      >台湾</option>
  </select>
</td>
```

```
</tr>
<tr>
    <td align="right">教育程度：</td>
    <td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
    <td colspan="2">
        <select name=education size=1>
            <option value="null" selected>请选 ...</option>
            <option value="Below high school">高中以下程度</option>
            <option value="high school/5-year college">高中/中专</option>
            <option value="College/University">大学/专科</option>
            <option value="Graduate and above">研究生以上</option>
        </select>
    </td>
</tr>
<tr>
<tr>
    <td align="right">兴趣爱好：</td>
    <td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
    <td>
        <input type=checkbox name=hobbies value="1@">
        计算机行业/软硬件
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
        <input type=checkbox name=hobbies value="2@">
        旅游、度假 </td>
</tr>
<tr>
<tr>
    <td align="right">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
    <td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
    <td>
        <input type=checkbox name=hobbies value="3@">
        上网聊天、游戏
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
        <input type=checkbox name=hobbies value="4@">
        阅读、图书音像</td>
</tr>
<tr>
<tr>
    <td align="right">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
    <td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
    <td>
        <input type=checkbox name=hobbies value="5,">
        文化/教育/育儿
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
        <input type=checkbox name=hobbies value="6,">
        情感生活 </td>
</tr>
<tr>
    <td align="right">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
    <td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
```

227



```
        window.alert("请填写密码!");
        window.form1.passwd1.focus();
        return false;
    }
    if(document.form1.passwd1.value!=document.form1.passwd2.value)
    {
        window.alert("请验证密码!");
        window.form1.passwd2.focus();
        return false;
    }
    if(document.form1.email.value=="")
    {
        window.alert("请填写您的 E_mail!");
        window.form1.email.focus();
        return false;
    }
    document.form1.submit();
}
</script>
</td>
</tr>
</form>

</table>
</body>
</html>
```

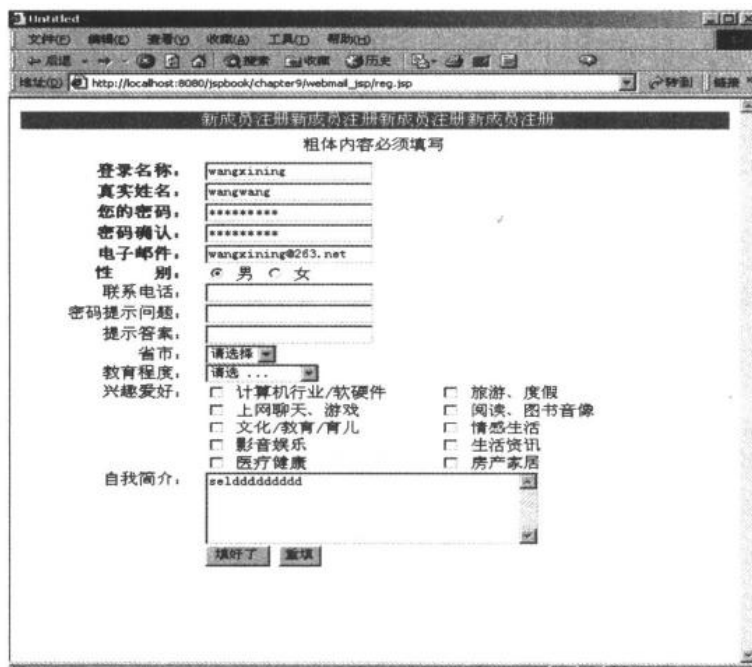


图 9-16 用户注册界面

如果注册成功,则会进入 main.jsp 的界面。反之,则会出现如图 9-17 所示的界面。

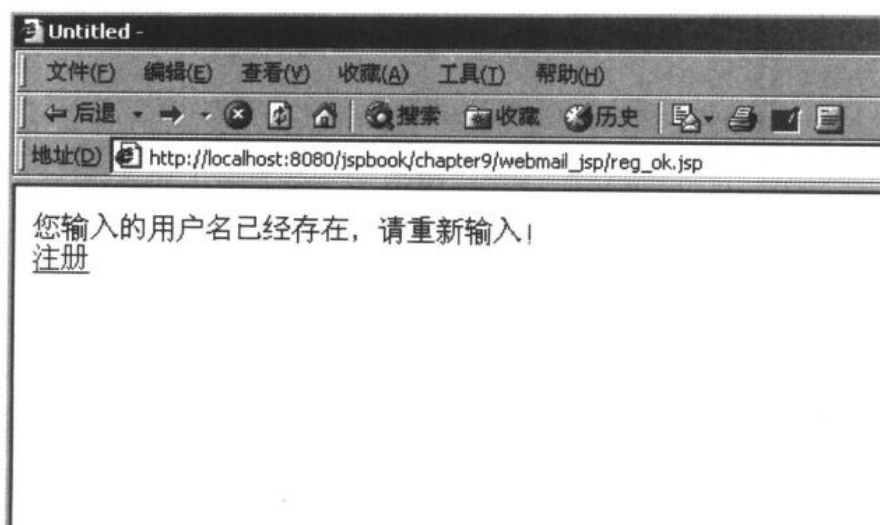


图 9-17 出错警告界面

出错界面 error.jsp 文件的源代码如例程 9-5 所示。

例程 9-5

```
<%@ page contentType="text/html;charset=gb239" %>
<%@page language="java" import="java.sql.*"%>
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>

<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>

<%!String logname,realname,passwd1,passwd2,email,gender,phone;
String problem,answer,province,education,selfintro,hobby;
String[] hobbies;
boolean regAttempt = false;
```

```
String errorMessage = "";
%>
<% //out.print(request.getParameterValues("logname"));
    //必须填写的项目
    logname=request.getParameter("logname");
    realname=request.getParameter("realname");
    passwd1=request.getParameter("passwd1");
    passwd2=request.getParameter("passwd2");
    email=request.getParameter("email");
    gender=request.getParameter("Gender");
    //非必须填写的项目
    phone=request.getParameter("phone");
    if(phone.trim().equals("")){
        phone=null;
    }
    problem=request.getParameter("problem");
    if(problem.trim().equals("")){
        problem=null;
    }
    answer=request.getParameter("answer");
    if(answer.trim().equals("")){
        answer=null;
    }
    province=request.getParameter("Province");
    if(province.trim().equals("")){
        province=null;
    }
    education=request.getParameter("education");
    if(education.trim().equals("")){
        education=null;
    }

    hobbies=request.getParameterValues("hobbies");
    hobby="";
    if(hobbies!=null){
        for (int i=0;i<hobbies.length;i++){
            hobby=hobby+hobbies[i];
        }
    }else hobby="null";
    selfintro=request.getParameter("selfintro");
    if(selfintro.trim().equals("")){
        selfintro=null;
    }

    ///转换中文
    logname=getStr(logname);
```

```

        realname=getStr(realname);
        passwd1 =getStr(passwd1);
        email=getStr(email);
        gender =getStr(gender);
        phone =getStr(phone);
        problem =getStr(problem);
        answer =getStr(answer);
        province =getStr(province);
        education=getStr(education);
        hobby=getStr(hobby);
        selfintro=getStr(selfintro);
    %>
<%
    String sql="select ID from member where logname='"+logname+"'";
    ResultSet RS=userBean.executeQuery(sql);
    out.println(sql);
    int rowcount=0;
    try
    {
        while(RS.next())
        {
            rowcount++;
        }
    }
    catch(Exception e)
    {
    }
    //count.....
    // out.println(rowcount);
    if(rowcount==0)
    {
        regAttempt=true;
    }else response.sendRedirect("error.jsp");
    if(regAttempt==true)
    {
        String sqlinsert="insert into member(logname,realname,password,email,gender,phone,
        problem,answer,province,education,hobbies,selfintro) Values('"+logname+"','"+realname+"','"+passwd1+"',
        '"+email+"','"+gender+"','"+phone+"','"+problem+"','"+answer+"','"+province+"','"+education+"','"+hobby+
        "','"+selfintro+"')";
        out.println(sqlinsert);
        userBean.executeQuery(sqlinsert);
        session.setAttribute("username",logname);
        response.sendRedirect("main.jsp");
    }
    %>

```

## 3. 查询密码界面（如图 9-18 所示）

findpass.jsp 文件的源代码如例程 9-6 所示。

例程 9-6

```

<%@ page contentType="text/html;charset=gb239" %>
<%@page language="java" import="java.sql.*"%>
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>
<html>
<head>
    <title>Untitled</title>
</head>
<body>
    <table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
        <tr>
            <td width="150" height="355" valign="top">

            </td>
            <td width="10" height="100%"></td>
            <td width="1" height="100%" bgcolor="#3399ff"></td>
            <td width="10" height="100%"></td>
            <td width="589" height="331" valign="top" background="images/bg1.gif">
                <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
                    <tr>
                        <td width="100%" colspan="2" height="20" bgcolor="#3399ff">&nbsp;<font
color="#ffffff">取回密码</font>
                    </td>
                </tr>
            </td>
        </tr>
    </table>

```

```
<form action="findpass_ok.jsp" method="post">
  <tr><td align="right" height="32" width="40%">登录名: </td>
  <td>
    <input type="text" name="logname" value="">
  </td>
</tr>
<tr>
  <td align="right" height="32">E_mail: </td>
  <td>
    <input type="text" name="email">
  </td>
</tr>
<tr> <td colspan="2" align="center">
  <input type="submit" value="找回密码" name="findpass">
</td>
</tr>
</form>

  <tr>
    <td colspan="2" height="150" align="right">&nbsp;  </td>
  </tr>

</table>
</td>
</tr>
</table>
</body>
</html>
```

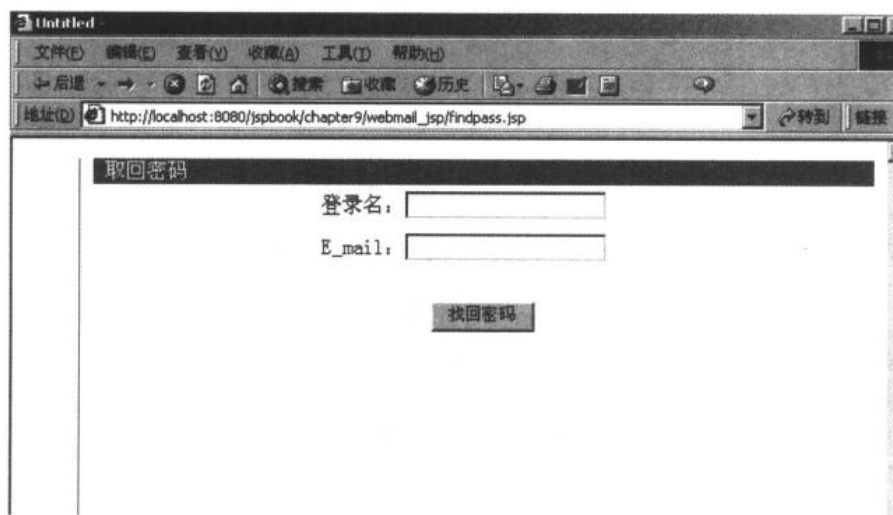


图 9-18 查询密码界面

如果输入的用户名和 email 不匹配，则会出现如图 9-19 所示的界面。



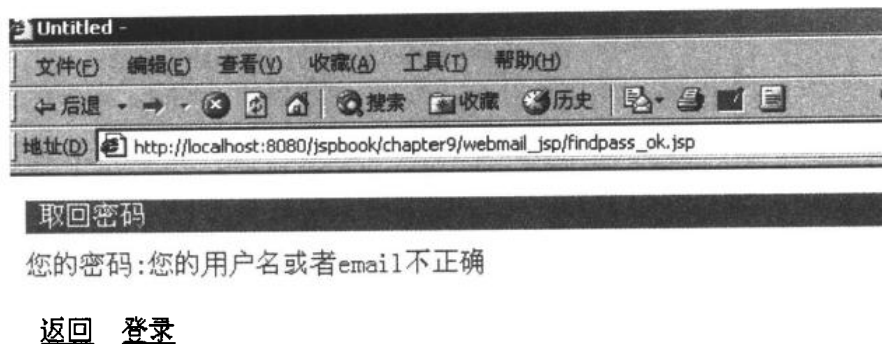


图 9-19 出错信息界面

如果输入的用户名和 email 相匹配, 则会出现如图 9-20 所示的界面。



图 9-20 取回密码界面

上图对应的 findpass\_ok.JSP 文件的源代码如例程 9-7 所示。

#### 例程 9-7

```
<%@ page contentType="text/html; charset=gb239" %>
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
```

```

        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {

    }
    return "null";
}
%>
<html>
<head>
    <title>Untitled</title>
</head>

<body>
    <table border="0" width="100%" cellspacing="0" cellpadding="0" height="">
        <tr>
            <td width="100%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff">取回
密码</font>
            </td>
        </tr>
    </table>

<%!
    String logname,email;
    boolean loginAttempt = false;
    String errorMessage = "";
    //logname = request.getParameter("logname");
    //    logpass = request.getParameter("logpass");

%>
<%
if(request.getParameterValues("findpass") != null
&&request.getParameterValues("logname") != null
&&request.getParameterValues("email") != null)
{
    loginAttempt = true;
}
if (loginAttempt)
{
    logname=request.getParameter("logname");
    email=request.getParameter("email");
    logname=getStr(logname);
    email=getStr(email);
    String sql="select * from member where logname='"+logname+"' and email='"+email+"'";
    //out.println(sql);

```

```

ResultSet RS=userBean.executeQuery(sql);

int rowcount=0;
try
{
    while(RS.next())
    {
        rowcount++;
        errorMessage=RS.getString("password");
    }
}
catch(Exception e)
{
}

if(rowcount!=0)
{
    //errorMessage=RS.getString("password");
}else errorMessage="您的用户名或者 email 不正确";
}
%>
<tr><td height="40" >
<font color=red>您的密码:<%=errorMessage%></font>
</td>
</tr>

<tr>
<td height="40" >&nbsp;<a href="findpass.jsp">返回</a>
&nbsp;<a href="login.jsp">登录</a>
</td>
</tr>

</table>
</td>
</tr>
</table>
</body>
</html>

```

#### 4. WebMail 主界面（如图 9-21 所示）

main.jsp 页面文件的源代码如例程 9-8 所示。

例程 9-8

```

<%
String userID;
userID=(String)session.getAttribute("username");
if(userID=="")!userID.equals(""))

```

```

{
    response.sendRedirect("login.jsp");
}
%>
<html>
    <head>
        <link rel="stylesheet" href="CssLib/CssForMenu.css">
        <meta http-equiv="Content-Type" content="text/html; charset=gb239">
        <title></title>
        <Script Language=JavaScript src = "menu/FunctionForMenu.js"></Script>
    </head>
    <body scroll="no" style="MARGIN: 0px">
        <table border="0" cellspacing="0" cellpadding="0" width="100%" height="100%" id =
        "tblTotal" Name = "tblTotal">
            <tr>
                <td id="frmMenu" name="frmMenu" nowrap valign="center" align="middle"
class=tblLeftbody>
                    <iframe id="BoardTitle" name=BoardTitle style="HEIGHT: 100%;
VISIBILITY: inherit; WIDTH: 150px; Z-INDEX: 2" frameborder="0" src="menu/Menu_
new.htm"></iframe>
                </td>
                <td style="WIDTH: 7pt" bgcolor="#336699" onclick="switchSysBar()"
class=tblbody>
                    <span class="navPoint" id="switchPoint" title="关闭/开启工具栏
">3</span>
                </td>
                <td width="100%">
                    <iframe id="frmRight" name="frmRight" style="HEIGHT: 100%;
VISIBILITY: inherit; WIDTH: 100%; Z-INDEX: 1" frameborder=0 src="welcome.jsp"
                    ></iframe>
                </td>
            </tr>
        </table>
    </body>
</html>

```

在这个页面中，我们使用 JavaScript 语言实现了动态菜单的功能。在这里就不再讲述这个 JavaScript 的实现方法了。代码可以在配套的光盘中找到。

main.jsp 页面是一个框架结构，左边是 menu/Menu\_new.htm，右边是 welcome.jsp 页面。

左边页面是一个动态菜单页面，右边页面是一个欢迎页面。

Welcome.jsp 页面文件的源代码如例程 9-9 所示。

例程 9-9

```

<%
String userID;

```

```
userID=(String)session.getAttribute("username");
if(userID=="")!userID.equals(""))
{
    response.sendRedirect("login.jsp");
}
%>
<html>
<head>
    <title>Untitled</title>
</head>

<body>
    <%=session.getAttribute("username")%>您好，欢迎使用 WebMail 系统。
</body>
</html>
```

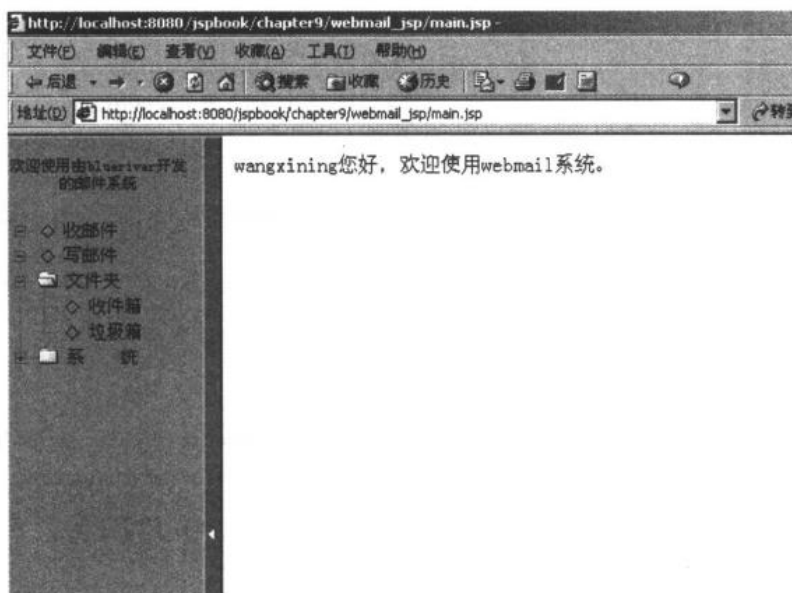


图 9-21 Web mail 主界面

## 5. 写邮件界面（如图 9-22 所示）

compose.jsp 页面文件的源代码如例程 9-10 所示。

例程 9-10

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb239">
<script Language="JavaScript" src="menu/style_plusminus/menu.js"></script>
</head>

<body bgcolor="#FFFFFF">
```

```
<form name="form1" method="post" action="send.jsp">
<table width="100%" border="1">
  <tr>
    <td width="30%">
      <div align="right">收件人: </div>
    </td>
    <td width="70%">
      <input type="text" name="toID" size="40">
    </td>
  </tr>
  <tr>
    <td width="30%">
      <div align="right">标题: </div>
    </td>
    <td width="70%">
      <input type="text" name="title" size="40">
    </td>
  </tr>
  <tr>
    <td width="30%">
      <div align="right">内容: </div>
    </td>
    <td width="70%">
      <textarea name="content" cols="80" rows="20"></textarea>
    </td>
  </tr>
  <tr>
    <td width="30%">
      <div align="right">附件: </div>
    </td>
    <td width="70%">
      <input type="text" name="fujian" size="40" readonly>
      <a href=javascript:openwin('upload.jsp')> 上传附件</a>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <div align="center">
        <input type="button" name="Submit" value="发送" onclick="chk()">
        <input type="reset" name="Submit2" value="重写">
      </div>
    </td>
  </tr>
</table>
<script language="javascript">
  function chk()
  {
    if(document.form1.toID.value=="")
    {
      window.alert("请填写收件人名!");
    }
  }
</script>
```



```
        window.form1.toID.focus();
        return false;
    }
    if(document.form1.title.value=="")
    {
        window.alert("请填写标题! ");
        window.form1.title.focus();
        return false;
    }
    if(document.form1.content.value=="")
    {
        window.alert("请填写内容! ");
        window.form1.content.focus();
        return false;
    }

    document.form1.submit();
}

</script>
</div>
</td>
</tr>
</table>
</form>
</body>
</html>
```

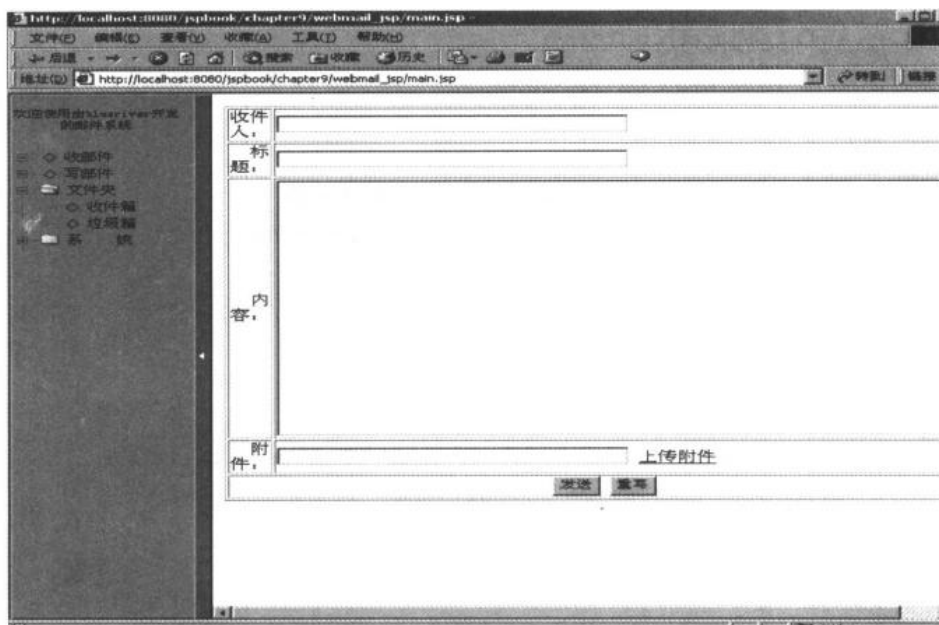


图 9-22 写邮件界面

在这个页面中，您可以上传附件。如图 9-23 所示。

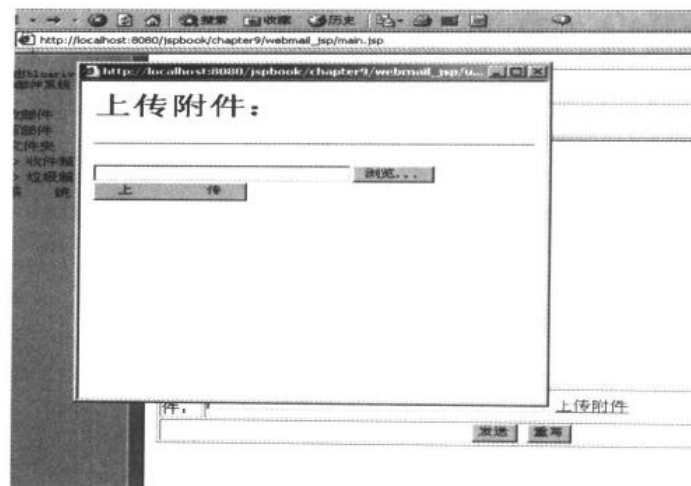


图 9-23 上传附件界面

upload.jsp 页面文件的源代码如例程 9-11 所示。

例程 9-11

```
<HTML>
<BODY BGCOLOR="white">
<H1>上传附件: </H1>
<HR>
<FORM METHOD="POST" ACTION="upload_ok.jsp" ENCTYPE="multipart/form-data">
  <INPUT TYPE="FILE" NAME="FILE1" SIZE="30"><BR>
  <INPUT TYPE="SUBMIT" VALUE="上      传">
</FORM>
</BODY>
</HTML>
```

选择文件上传，如图 9-24 所示。

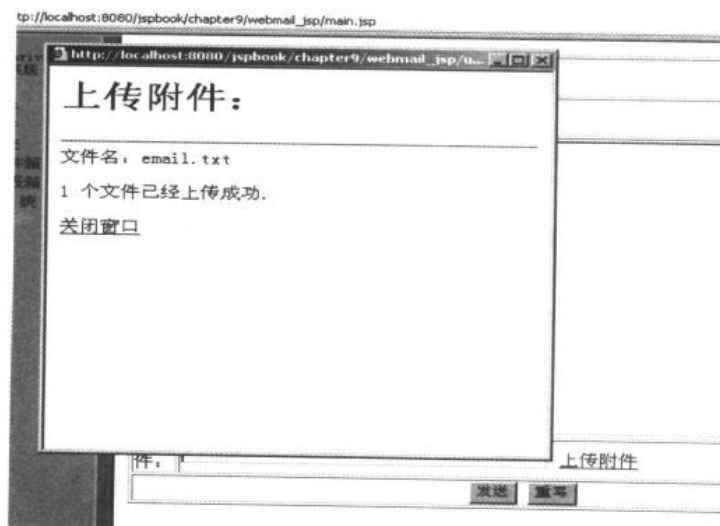


图 9-24 附件上传界面

upload\_ok.jsp 页面文件的源代码如例程 9-12 所示。

例程 9-12

```
<%@ page contentType="text/html;charset=gb239" %>
<%@ page language="java" import="com.jspsmart.upload.*"%>
<jsp:useBean id="mySmartUpload" scope="page" class="com.jspsmart.upload.SmartUpload" />

<HTML>
<BODY BGCOLOR="white">
<H1>上传附件: </H1>
<HR>

<%
    // Variables
    int count=0;
    // Initialization
    mySmartUpload.initialize(pageContext);
    mySmartUpload.setTotalMaxFileSize(1000000000);
    // Upload
    mySmartUpload.upload();
    try {
        // Save the files with their original names in the virtual path "/upload"
        // if it doesn't exist try to save in the physical path "/upload"
        count = mySmartUpload.save("/WebMail_jsp/upload");

        com.jspsmart.upload.File myFile = mySmartUpload.getFiles().getFile(0);
        String fileName=myFile.getFileName();
        out.println("文件名: "+fileName+"<br><br>");
        // Display the number of files uploaded
        out.println(count + " 个文件已经上传成功.<br><br>");
    }
    %>
<a href="javascript:fujianName()">关闭窗口</a>
<script language="javascript">
<!--
function fujianName()
{
    opener.document.form1.fujian.value="<%=fileName%>";
    self.close();
}
-->
</script>
<%
    } catch (Exception e)
        out.println(e.toString());
    }
    %>
```

&lt;/BODY&gt;

&lt;/HTML&gt;

在图 9-25 中填写完所有的必填信息后,单击【发送】按钮发送邮件。

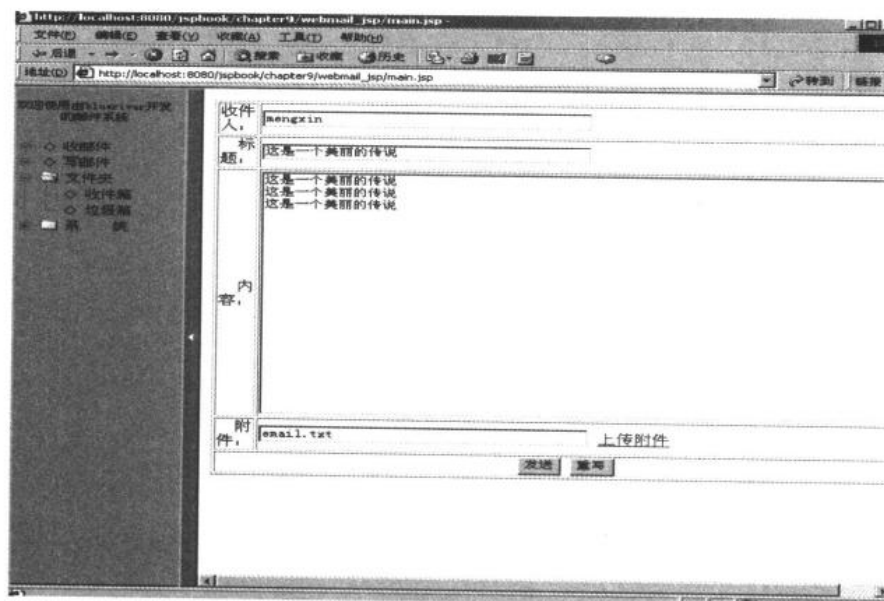


图 9-25 写邮件界面

发送邮件后的界面如图 9-26 所示。

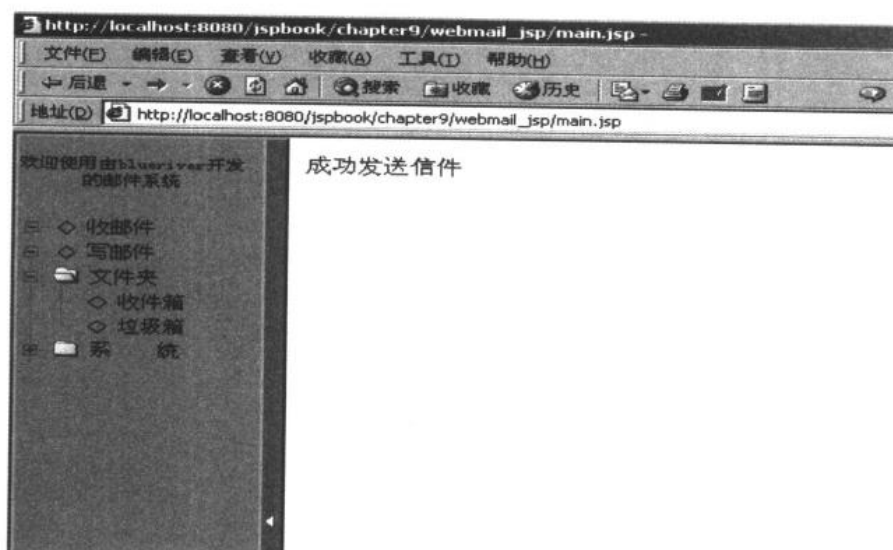


图 9-26 邮件发送成功界面

send.jsp 页面文件的源代码如例程 9-13 所示。

#### 例程 9-13

```
<%@ page contentType="text/html; charset=gb239" %>
```

```
<%@page language="java" import="java.sql.*"%>
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb239">
<script Language="JavaScript" src="menu/style_plusminus/menu.js"></script>
</head>
<body bgcolor="#FFFFFF">
<%
String toID,title,content,fujian;
toID=request.getParameter("toID");
title=request.getParameter("title");
content=request.getParameter("content");
fujian=request.getParameter("fujian");

//转换成中文
toID=getStr(toID);
title=getStr(title);
content=getStr(content);
fujian=getStr(fujian);

String sql;
sql="insert into inbox(userID,fromID,title,content,fujian) values(";
sql=sql+toID+", '"+session.getAttribute("username")+"','"+title+"',";
sql=sql+" '"+content+"','"+fujian+"')";
try
{
    userBean.executeQuery(sql);
}
```

```
}  
catch(Exception e)  
{  
    out.println(e);  
}  
%>  
成功发送信件  
</body>  
</html>
```

## 6. 收邮件界面（如图 9-27 所示）

mail\_List.jsp 页面文件的源代码如例程 9-14 所示。

例程 9-14

```
<%@ page contentType="text/html;charset=gb239" %>  
<%@page language="java" import="java.sql.*"%>  
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>  
<%!  
public String getStr(String str)  
{  
    try  
    {  
        String temp_p=str;  
        byte[] temp_t=temp_p.getBytes("ISO8859-1");  
        String temp=new String(temp_t);  
        return temp;  
    }  
    catch(Exception e)  
    {  
  
    }  
    return "null";  
}  
%>  
<html>  
<head>  
<title>Untitled Document</title>  
<meta http-equiv="Content-Type" content="text/html; charset=gb239">  
</head>  
  
<body bgcolor="#FFFFFF">  
<table width="100%" border="1">  
    <tr>  
        <td width="18%">发件人</td>  
        <td width="25%">标题</td>  
        <td width="29%">时间</td>  
        <td width="28%">附件</td>
```



```
</tr>
<%
String sql;
sql="select * from inbox where userID='"+session.getAttribute("username")+"'";
ResultSet RS;
RS=userBean.executeQuery(sql);
while(RS.next())
{
int id;
id=RS.getInt("id");
%>
<tr>
<td width="18%">&nbsp;<%=RS.getString("fromID")%></td>
width="25%">&nbsp;<a
href="read.jsp?id=<%=id%>"><%=RS.getString("title")%></a></td>
<td width="29%">&nbsp;<%=RS.getDate("send_time")%></td>
<td width="28%">&nbsp;<%=RS.getString("fujian")%></td>
</tr>
<%
}
RS.close();
%>
</table>
</body>
</html>
```

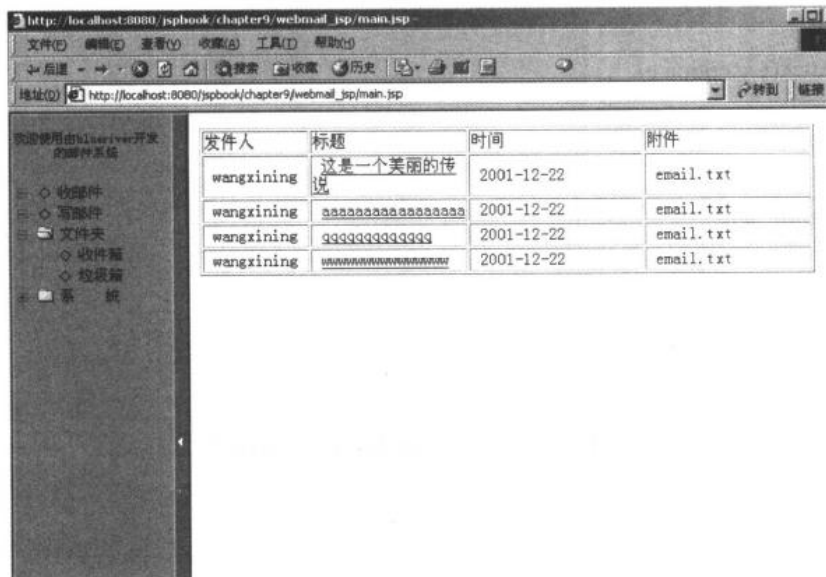


图 9-27 收邮件界面

单击标题，则可以阅读邮件信息，如图 9-28 所示。

Read.jsp 页面文件的源代码如例程 9-15 所示。

例程 9-15

```
<%@ page contentType="text/html;charset=gb239" %>
<%@ page language="java" import="java.sql.*"%>
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb239">
</head>
<body bgcolor="#FFFFFF">
<%
String id;
id=request.getParameter("id");
String sql;
sql="select * from inbox where id="+id;
ResultSet RS;
RS=userBean.executeQuery(sql);
if(RS.next())
{
%>
<table width="75%" border="1">
<tr>
<td width="30%">
<div align="right">发件人: </div>
</td>
<td width="70%">&nbsp;<%=RS.getString("fromID")%> </td>
</tr>
<tr>
<td width="30%">
```

```
<div align="right">标题: </div>
</td>
<td width="70%">&nbsp;   <%=RS.getString("title")%> </td>
</tr>
<tr>
<td width="30%">
<div align="right">内容: </div>
</td>
<td width="70%">&nbsp;   <%=RS.getString("content")%> </td>
</tr>
<tr>
<td width="30%">
<div align="right">附件: </div>
</td>
<%
String fj;
fj=RS.getString("fujian");
%>
<td width="70%">&nbsp;   <a href="upload/<%=fj%>"><%=fj%></a> </td>
</tr>
<tr>
<td colspan="2">
<div align="center"></div>
</td>
</tr>
</table>
<%
}
RS.close();
%>
</body>
</html>
```

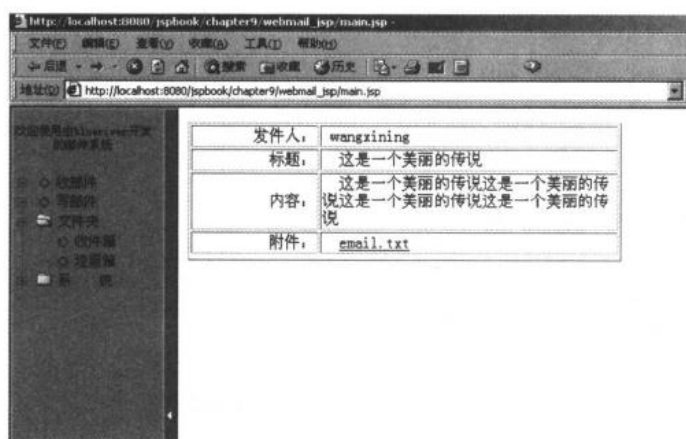


图 9-28 阅读邮件界面

## 7. 个人密码修改界面（如图 9-29 所示）

editpass.jsp 页面文件的源代码如例程 9-16 所示。

例程 9-16

```
<%@ page contentType="text/html;charset=gb239" %>
<%@page language="java" import="java.sql.*"%>
<jsp:useBean id="userBean" scope="page" class="WebMail.conn"/>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {

    }
    return "null";
}
%>
<html>
<head>
    <title>Untitled</title>
</head>
<body>
    <table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
        <tr>
            <td width="150" height="355" valign="top">

            </td>
            <td width="10" height="100%"></td>
            <td width="1" height="100%" bgcolor="#3399ff"></td>
            <td width="10" height="100%"></td>
            <td width="589" height="331" valign="top" background="images/bg1.gif">
                <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
                    <tr>
                        <td width="100%" colspan="2" height="20" bgcolor="#3399ff">&nbsp;&nbsp;&nbsp;<font
color="#ffffff">修改密码</font>
                    </td>
                </tr>
            </td>
        </tr>
    </table>
</body>
</html>
```

```
<form action="editpass_ok.jsp" method="post">
  <tr><td align="right" height="32" width="40%">登录名: </td>
    <td>
      <%=session.getAttribute("username")%>
    </td>
  </tr>
  <tr>
    <td align="right" height="32">旧密码: </td>
    <td>
      <input type="text" name="oldpass">
    </td>
  </tr>

  <tr>
    <td align="right" height="32">新密码: </td>
    <td>
      <input type="text" name="newpass">
    </td>
  </tr>

  <tr>
    <td align="right" height="32">确认新密码: </td>
    <td>
      <input type="text" name="cfmnewpass">
    </td>
  </tr>
  <tr> <td colspan="2" align="center">
    <input type="submit" value="修改密码" name="editpass">
  </td>
  </tr>
</form>

  <tr>
    <td colspan="2" height="150" align="right">&nbsp;  </td>
  </tr>

</table>
</td>
</tr>
</table>
</body>

</html>
```

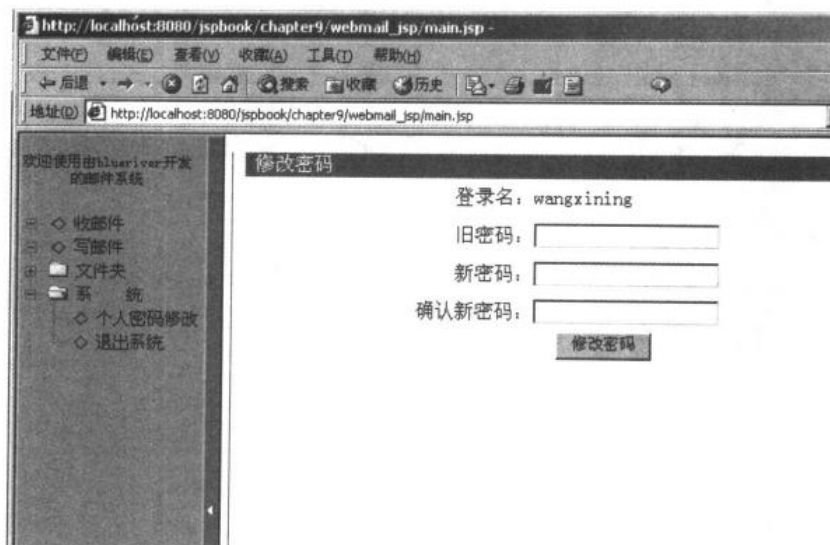


图 9-29 个人密码修改界面

输入旧密码、新密码和确认新密码后，单击【修改密码】按钮。如果输入的新密码和确认新密码不匹配，则会出现如图 9-30 所示的界面。

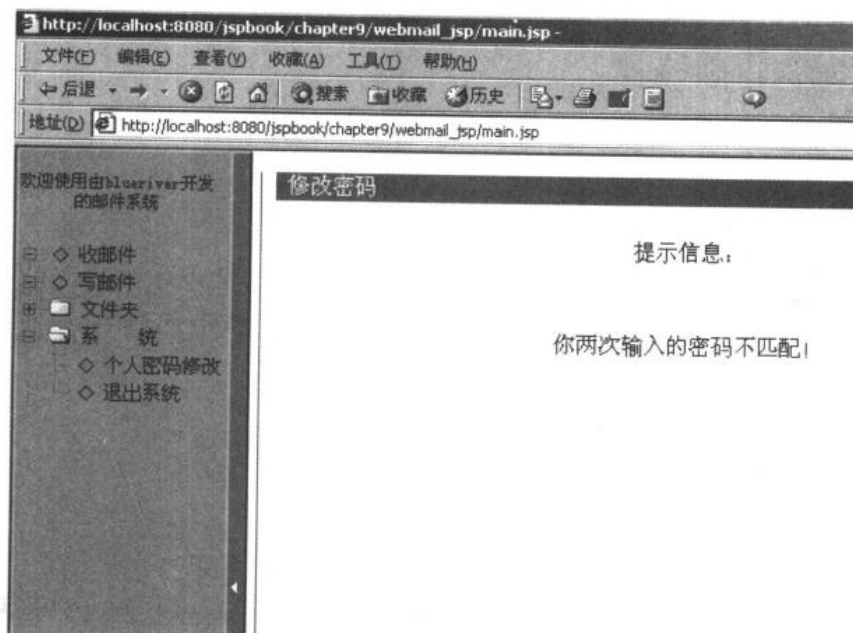


图 9-30 密码错误界面

如果两次输入的新密码一致，则会成功地修改密码。界面如图 9-31 所示。



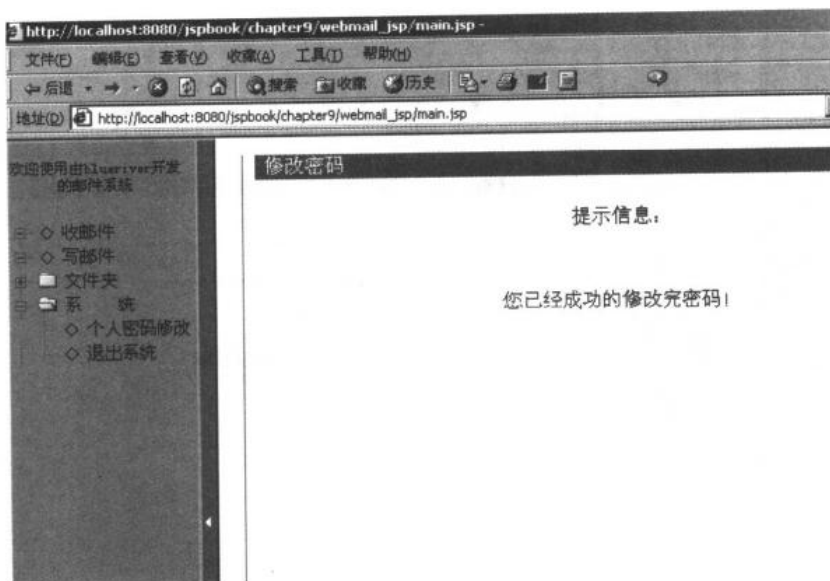


图 9-31 密码正确界面

## 8. 退出系统

logout.jsp 页面文件源代码如下:

```
<html>
<head>
    <title>Untitled</title>
</head>

<body>
<%
    session.setAttribute("username","");
    response.sendRedirect("login.jsp");
%>
</body>
</html>
```

### 9.2.2 使用模式二实现

#### 1. 用户登录界面

login.jsp 文件与使用模式一实现时的 login.jsp 文件代码基本相同, 只是两者提交的页面不同。两者的区别如下:

- 使用模式一实现时的提交  
`<form action="login_ok.jsp" method="post">`
- 使用模式二实现时的提交  
`<form action="/test/servlet/WebMailServlet.login" method="post">`

WebMailServlet.login 源代码如例程 9-17 所示。

例程 9-17

```
package WebMailervlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class login extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:WebMail";
        Connection connect = null;
        ResultSet rs = null;
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
            return temp;
        }
        catch(Exception e)
        {

```

```
e.printStackTrace();
    }
    return "null";
}
/**Process the HTTP Get request*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<font color=\"green\">");
    out.println("<p>The servlet get method is the reply.</p>");
    out.println("</font>");

}
/**Process the HTTP Post request*/
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>login</title></head>");
    out.println("<body>");
    out.println("</body></html>");
    String logname,logpass;
    boolean loginAttempt = false;
    boolean loginOK = false;
    String errorMessage = "请您登录";
    HttpSession session=request.getSession(true);
    if(request.getParameterValues("login") != null
    &&request.getParameterValues("logname") != null
    &&request.getParameterValues("logpass") != null)
    {
        loginAttempt = true;
    }
    if (loginAttempt)
    {
        logname=request.getParameter("logname");
        logpass=request.getParameter("logpass");
        logname=getStr(logname);
        logpass=getStr(logpass);
        String sql="select * from member where logname='"+logname+"' and password='"+logpass+"'";
        out.println(sql);
        ResultSet RS=executeQuery(sql);
        int rowcount=0;
        try
        {
```

```
        while(RS.next())
        {
            rowscount++;
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    //count.....
    out.println(rowscount);
    if(rowscount!=0)
    {
        errorMessage="成功登录";
        session.setAttribute("username",logname);
        loginOK=true;

        if(loginOK){
            response.sendRedirect("../WebMail_sev/main.jsp");
        }
    }else{
        errorMessage="loginerr";
        session.setAttribute("username","");
        response.sendRedirect("../WebMail_sev/login.jsp?err="+errorMessage);
    }
}
}
/**Clean up resources*/
public void destroy() {
}
}
```

## 2. 用户注册界面

reg.jsp 文件与使用模式一实现时的 reg.jsp 文件代码基本相同，只是两者提交的页面不同。两者的区别如下：

- 使用模式一实现时的提交  
<form action=" reg\_ok.jsp" method="post">
- 使用模式二实现时的提交  
<form action="/test/servlet/WebMailServlet.reg " method="post">

WebMailServlet.reg 源代码如例程 9-18 所示。

例程 9-18

```
package WebMailServlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
import java.util.*;
import java.sql.*;

public class reg extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:WebMail";
        Connection connect = null;
        ResultSet rs = null;
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
            return temp;
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        return "null";
    }
}
/**Process the HTTP Get request*/
```

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>reg</title></head>");
    out.println("<body>");
    out.println("<p>The servlet has received a GET. This is the reply.</p>");
    out.println("</body></html>");
}

/**Process the HTTP Post request*/
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>reg</title></head>");
    out.println("<body>");
    String logname,realname,passwd1,passwd2,email,gender,phone;
    String problem,answer,province,education,selfintro,hobby;
    String[] hobbies;
    boolean regAttempt = false;
    String errorMessage = "";
    HttpSession session=request.getSession(true);
    //必须填写的项目
    logname=request.getParameter("logname");
    realname=request.getParameter("realname");
    passwd1=request.getParameter("passwd1");
    passwd2=request.getParameter("passwd2");
    email=request.getParameter("email");
    gender=request.getParameter("Gender");
    //非必须填写的项目
    phone=request.getParameter("phone");
    if(phone.trim().equals("")){
        phone=null;
    }
    problem=request.getParameter("problem");
    if(problem.trim().equals("")){
        problem=null;
    }
    answer=request.getParameter("answer");
    if(answer.trim().equals("")){
        answer=null;
    }
    province=request.getParameter("Province");
    if(province.trim().equals("")){
```



```
        province=null;
    }
    education=request.getParameter("education");
    if(education.trim().equals("")){
        education=null;
    }

    hobbies=request.getParameterValues("hobbies");
    hobby="";
    if(hobbies!=null){
        for (int i=0;i<hobbies.length;i++){
            hobby=hobby+hobbies[i];
        }
    }else hobby="null";
    selfintro=request.getParameter("selfintro");
    if(selfintro.trim().equals("")){
        selfintro=null;
    }

    ///转换中文
    logname=getStr(logname);
    realname=getStr(realname);
    passwd1 =getStr(passwd1);
    email=getStr(email);
    gender =getStr(gender);
    phone =getStr(phone);
    problem =getStr(problem);
    answer =getStr(answer);
    province =getStr(province);
    education=getStr(education);
    hobby=getStr(hobby);
    selfintro=getStr(selfintro);

    String sql="select ID from member where logname='"+logname+"'";
    ResultSet RS=executeQuery(sql);
    out.println(sql);
    int rowcount=0;
    try
    {
        while(RS.next())
        {
            rowcount++;
        }
    }
    catch(Exception e)
    {
```

```

    }
    //count.....
    // out.println(rowcount);
    if(rowcount==0)
    {
        regAttempt=true;
    }else response.sendRedirect("../WebMail_sev/error.jsp");
    if(regAttempt==true)
    {
        String sqlinsert="insert into
member(logname,realname,password,email,gender,phone,problem,answer,province,education,hobbies,selfi
ntro)
Values('"+logname+"','"+realname+"','"+passwd+"','"+email+"','"+gender+"','"+phone+"','"+problem+"','"+
answer+"','"+province+"','"+education+"','"+hobby+"','"+selfintro+"')";
        out.println(sqlinsert);
        executeQuery(sqlinsert);
        session.setAttribute("username",logname);
        response.sendRedirect("../WebMail_sev/main.jsp");
    }

    out.println("</body></html>");
}
/**Clean up resources*/
public void destroy() {
}
}

```

### 3. 查询密码界面

reg.jsp 文件与使用模式一实现时的 reg.jsp 文件代码基本相同，只是两者提交的页面不同。两者的区别如下：

- 使用模式一实现时的提交  
<form action=" findpass \_ok.jsp" method="post">
- 使用模式二实现时的提交  
<form action="/test/servlet/WebMailServlet. findpwd " method="post">

WebMailServlet. findpwd 源代码如例程 9-19 所示。

例程 9-19

```

package WebMailServlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
public class findpwd extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";

```

```
/**Initialize global variables*/
public void init() throws ServletException {
    String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
    try {
        Class.forName(sDBDriver);
    }
    catch(java.lang.ClassNotFoundException e) {
        System.err.println( e.getMessage());
    }
}

public ResultSet executeQuery(String sql) {
    String sConnStr = "jdbc:odbc:WebMail";
    Connection connect = null;
    ResultSet rs = null;
    rs = null;
    try {
        connect = DriverManager.getConnection(sConnStr);
        Statement stmt = connect.createStatement();
        rs = stmt.executeQuery(sql);
    }
    catch(SQLException ex) {
        System.err.println(ex.getMessage());
    }
    return rs;
}

public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}

/**Process the HTTP Get request*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
}
```

```

        out.println("<head><title>findpwd</title></head>");
        out.println("<body>");
        out.println("<p>The servlet has received a GET. This is the reply.</p>");
        out.println("</body></html>");
    }
    /**Process the HTTP Post request*/
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>findpwd</title></head>");
        out.println("<body>");
        out.println(" <table border=0 width=100% cellpadding=0 height=> ");
        out.println(" <tr> ");
        out.println("<td width=100% height=20 bgcolor=#3399ff>&nbsp;  <font color=#ffffff>取回密码
</font>");
        out.println("</td>");
        out.println("</tr>");
        String logname,email;
        boolean loginAttempt = false;
        String errorMessage = "";
        if(request.getParameterValues("findpass") != null
&&request.getParameterValues("logname") != null
&&request.getParameterValues("email") != null)
        {
            loginAttempt = true;
        }
        if (loginAttempt)
        {
            logname=request.getParameter("logname");
            email=request.getParameter("email");
            logname=getStr(logname);
            email=getStr(email);
            String sql="select * from member where logname='"+logname+"' and email='"+email+"'";
            //out.println(sql);
            ResultSet RS=executeQuery(sql);

            int rowcount=0;
            try
            {
                while(RS.next())
                {
                    rowcount++;
                    errorMessage=RS.getString("password");
                }
            }
        }
    }

```

```
    }
    catch(Exception e)
    {

    }

    if(rowcount!=0)
    {
        //errorMessage=RS.getString("password");
    }else errorMessage="您的用户名或者 email 不正确";
}

out.println("<tr><td height=40 >");
out.println("<font color=red>您的密码:");
    out.println(errorMessage);
    out.println("</font>");
out.println("</td>");
out.println("</tr>");
out.println("<tr >");
out.println("<td height=40 >&nbsp;  <a href=../WebMail_sev/findpass.jsp>返回</a>");
out.println("&nbsp;  <a href=../WebMail_sev/login.jsp>登录</a>");
out.println("</td>");
out.println("</tr>");
out.println("</table> ");
    out.println("</td>");
    out.println("</tr>");
    out.println("</table>");
out.println("</body></html>");

}

/**Clean up resources*/
public void destroy() {
}

}
```

#### 4. WebMail 主界面

在主界面中，页面的代码基本上跟使用模式一实现时的页面相同，只是有关的链接随之改变，因为链接地址不再是 jsp 文件，而是 servlet 文件。

具体的代码在这里不再叙述，请参考配套光盘。

#### 5. 写邮件界面

compose.jsp 文件与使用模式一实现时的 compose.jsp 文件代码基本相同，只是两者提交的页面不同。两者的区别如下：

- 使用模式一实现时的提交  
<form name="form1" method="post" action="send.jsp">
- 使用模式二实现时的提交  
<form action="/test/servlet/WebMailServlet.send" method="post">

WebMailServlet.send 源代码如例程 9-20 所示。

例程 9-20

```
package WebMailServlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class send extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:WebMail";
        Connection connect = null;
        ResultSet rs = null;
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
            return temp;
        }
    }
}
```



```
        catch(Exception e)
        {

        }

        return "null";
    }

    /**Process the HTTP Get request*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>send</title></head>");
        out.println("<body>");
        out.println("<p>The servlet has received a GET. This is the reply.</p>");
        out.println("</body></html>");
    }

    /**Process the HTTP Post request*/
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        HttpSession session=request.getSession(true);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>send</title></head>");
        out.println("<body>");
        String toID,title,content,fujian;
        toID=request.getParameter("toID");
        title=request.getParameter("title");
        content=request.getParameter("content");
        fujian=request.getParameter("fujian");

        //转换成中文
        toID=getStr(toID);
        title=getStr(title);
        content=getStr(content);
        fujian=getStr(fujian);

        String sql;
        sql="insert into inbox(userID,fromID,title,content,fujian) values(";
        sql=sql+toID+"",""+session.getAttribute("username")+",""+title+"","";
        sql=sql+""+content+"",""+fujian+"");
        out.println(sql);
        try
        {
            executeQuery(sql);
        }
    }
}
```

```
}  
catch(Exception e)  
{  
    out.println(e);  
}  
    out.println("成功发送信件");  
    out.println("</body></html>");  
}  
/**Clean up resources*/  
public void destroy() {  
}  
}
```

## 6. 收邮件界面

用 servlet 技术实现的收邮件的 WebMailServlet. Mailist 文件的源代码如例程 9-21 所示。

例程 9-21

```
package WebMailServlet;  
import javax.servlet.*;  
import javax.servlet.http.*;  
import java.io.*;  
import java.util.*;  
import java.sql.*;  
  
public class mailist extends HttpServlet {  
    private static final String CONTENT_TYPE = "text/html; charset=GBK";  
    /**Initialize global variables*/  
    public void init() throws ServletException {  
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";  
        try {  
            Class.forName(sDBDriver);  
        }  
        catch(java.lang.ClassNotFoundException e) {  
            System.err.println( e.getMessage());  
        }  
    }  
    public ResultSet executeQuery(String sql) {  
        String sConnStr = "jdbc:odbc:WebMail";  
        Connection connect = null;  
        ResultSet rs = null;  
        rs = null;  
        try {  
            connect = DriverManager.getConnection(sConnStr);  
            Statement stmt = connect.createStatement();  
            rs = stmt.executeQuery(sql);  
        }  
    }  
}
```

```
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
            return temp;
        }
        catch(Exception e)
        {
        }
        return "null";
    }
}
/**Process the HTTP Get request*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    HttpSession session=request.getSession(true);
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Untitled Document</title>");
    out.println("<meta http-equiv=\"Content-Type content=text/html; charset=gb239\">");
    out.println("</head>");

    out.println("<body bgcolor=#FFFFFF>");
    out.println("<table width=100% border=1>");
        out.println("<tr>");
            out.println(" <td width=18%>发件人</td>");
            out.println(" <td width=25%>标题</td>");
            out.println(" <td width=29%>时间</td>");
            out.println(" <td width=28%>附件</td>");
        out.println(" </tr>");
        String sql;
        sql="select * from inbox where userID='"+session.getAttribute("username")+"'";
        ResultSet RS;
        RS=executeQuery(sql);
        try
        {
```

```

while(RS.next())
{
    int id;
    id=RS.getInt("id");
    out.println("<tr>");
    out.println("<td width=\"18%\">&nbsp;");
    out.println(RS.getString("fromID"));
    out.println("</td>");
    out.println("<td width=\"25%\">&nbsp;");
    out.println("<a href=\"../WebMail_sev/read.jsp?id=\"");
    out.println(id);
    out.println("\>");
    out.println(RS.getString("title"));
    out.println("</a></td>");
    out.println("<td width=\"29%\">&nbsp;");
    out.println(RS.getDate("send_time"));
    out.println("</td>");
    out.println("<td width=\"28%\">&nbsp;");
    out.println(RS.getString("fujian"));
    out.println("</td>");
    out.println("</tr>");
}
RS.close();
}
catch(Exception e)
{
    System.err.println(e.getMessage());
}

out.println("</table>");
out.println("</body>");
out.println("</html>");
}
/**Clean up resources*/
public void destroy() {
}
}

```

## 7. 个人密码修改界面

editpass.jsp 文件与使用模式一实现时的 editpass.jsp 文件代码基本相同，只是两者提交的页面不同。两者的区别如下：

- 使用模式一实现时的提交  
<form action="editpass\_ok.jsp" method="post">
  - 使用模式二实现时的提交  
<form action="/test/servlet/WebMailServlet.editpwd " method="post">
- WebMailServlet.editpwd 源代码如例程 9-22 所示。

## 例程 9-22

```
package WebMailServlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class editpwd extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {

        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:WebMail";
        Connection connect = null;
        ResultSet rs = null;
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
            return temp;
        }
        catch(Exception e)
    }
```

```

    {

    }
    return "null";
}
/**Process the HTTP Get request*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>editpwd</title></head>");
    out.println("<body>");
    out.println("<p>The servlet has received a GET. This is the reply.</p>");
    out.println("</body></html>");
}
/**Process the HTTP Post request*/
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    HttpSession session=request.getSession(true);
    out.println("<html>");
    out.println("<head><title>editpwd</title></head>");
    out.println("<body>");
    out.println("<table align=center border=0 width=760 cellpadding=0
height=355>");
    out.println("<tr> ");
    out.println("<td width=150 height=355 valign=top> ");

    out.println("</td> ");
    out.println("<td width=10 height=100%></td>");
    out.println("<td width=1 height=100% bgcolor=#3399ff></td>");
    out.println("<td width=10 height=100%></td>");
    out.println("<td width=589 height=331 valign=top background=images/bg1.gif> ");
    out.println("<table border=0 width=100% cellpadding=0 height=307> ");
    out.println("<tr>");
    out.println("<td width=100% colspan=2 height=20 bgcolor=#3399ff>&nbsp;  <font
color=#ffffff>修改密码</font>");
    out.println("</td>");
    out.println("</tr>");

    out.println("<tr> <td colspan=2 align=center>");
    out.println("提示信息: ");

```



```
        out.println(" </td>");
        out.println("</tr>");
        out.println(" <tr> <td colspan=2 align=center>");

        String userID,oldPass,newPass,cfmnewpass;
        userID=(String)session.getAttribute("username");
        oldPass=request.getParameter("oldpass");
        newPass=request.getParameter("newpass");
        cfmnewpass=request.getParameter("cfmnewpass");

        String sql,pwd,errorMsg;
        pwd="";
        errorMsg="";
        boolean temp;
        ResultSet RS;
        sql="select * from member where logname='"+userID+"'";
        RS=executeQuery(sql);
        try
        {
            if(RS.next())
            {
                pwd=RS.getString("password");
                if((pwd==oldPass)||(pwd.equals(oldPass)))
                {
                    temp=true;
                }
                else
                {
                    errorMsg="您输入的旧密码不对！";
                    temp=false;
                }
            }
        }
        else
        {
            errorMsg="用户名不存在！";
        }
    }
    catch(Exception e)
    {
        System.err.println(e.getMessage());
    }
    if((newPass==cfmnewpass)||(newPass.equals(cfmnewpass)))
    {
        temp=true;
    }
    else
```

```

{
    errorMsg="您两次输入的密码不匹配! ";
    temp=false;
}
if(temp)
{
    String sqlEdit;
    sqlEdit="update member set password='"+newPass+"' where logname='"+userID+"'";
    executeQuery(sqlEdit);
    errorMsg="您已经成功的修改完密码! ";
}
out.println(errorMsg);
        out.println("</td>");
        out.println("</tr>");
out.println("<tr> ");
        out.println(" <td colspan=2 height=150 align=right>&nbsp;  </td>");
        out.println("</tr>");
        out.println("</table>      ");
out.println("</td>");
        out.println("</tr>");
out.println("</table>");
        out.println("</body></html>");
    }
    /**Clean up resources*/
    public void destroy() {
    }
}

```

## 8. 退出系统

WebMailServlet.logout 源代码如例程 9-23 所示。

例程 9-23

```

package WebMailServlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class logout extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {
    }
    /**Process the HTTP Get request*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();

```

```
out.println("<html>");
out.println("<head><title>logout</title></head>");
out.println("<body>");
out.println("<p>退出系统</p>");
out.println("</body></html>");
HttpSession session=request.getSession(true);
session.setAttribute("username","");
response.sendRedirect("../WebMail_sev/login.jsp");
}
/**Clean up resources*/
public void destroy() {
}
}
```

我们使用两种不同的模式来实现了同一个功能，所以在模式二中就没有再显示图片。读者可以通过详细阅读这两种模式实现的代码，从而熟悉两者的区别和联系，加深了对模式二优势的了解。

### 9.3 JSP 发信的工作原理

电子邮件消息传递系统在项目开发过程中，有着重要的作用。JavaMail 提供了一种方法与电子邮件信息系统进行交互。

使用 JavaMail，则可以通过 Internet 发送或接收电子邮件信息。JavaMail 依赖于 Java 激活框架（Java Activated Frame）。

在讲述 JavaMail 之前，我们先来讲述一下电子邮件信息系统所涉及到的基本概念。

我们先来讲述一下什么是电子邮件信息以及电子邮件信息的结构。

电子邮件信息包括一个信息头和一个信息体。

电子邮件的信息头是基于文本的名字和对应的值，他们标识了一个消息的特征，例如：收件人的电子邮件地址、发送人的发送邮件地址、信件的主题以及信件的内容等。大多数的电子邮件地址为以下格式：user\_name@mail\_server\_name.do-main\_name。例如：WangXiNing@btamail.net.cn。如果一个邮件服务器的主机名没有提供，DNS 将会引用与域名相关联的 DNS 属性以决定电子邮件信息应当被传送的特定的邮件服务器地址。

电子邮件的信息体是由 MIME（Multipurpose Internet Mail Extensions）定义的。MIME 消息体有多种形式，例如：文本格式、视频格式、图像格式以及声音格式等。MIME 为信息体定义了一组标准的头，包括任何二进制数据的内容编码格式和内容类型字段。

电子邮件消息传递系统可以由多种底层协议来实现，目前在业界最为流行的是由 Internet 协议簇（Suite）定义的协议。整个电子邮件消息传递系统的过程大致是这样的，首先是由电子邮件客户使用标准的 Internet 传输协议发送给有效的收件人地址，然后电子邮件被存放在消息仓库（Store）中，最后电子邮件客户在另外标准的 Internet 传输协议的作用下检索出电子邮件。

下面，我们来讲述一下在电子邮件信息系统中的有关基本协议、工作机制和服务等。

首先, 我们来讲述一下简单邮件传输协议。

简单邮件传输协议 SMTP (Simple Mail Transfer Protocol) 是一个运行在 TCP/IP 上的协议。SMTP 主要用来发送和接收电子邮件信息, 是一个信息传递的传输协议。SMTP 服务器的默认端口号是 25, 也就是说 SMTP 服务器在默认端口号 25 上监听。SMTP 客户使用一组简单的基于文本的命令与 SMTP 服务器进行通信。

SMTP 在建立了一个连接后, 为了接收响应, SMTP 客户首先发出一个命令来标识他们的电子邮件地址。如果 SMTP 服务器接受了发送者和接收者的标识命令, 它会利用一个 OK 响应确认每一个命令。客户发送的另一个命令意味着电子邮件信息体的开始, 信息体则以一个圆点终止。

SMTP 本身不能够运行排队机制, 它经常与其他的信息传递服务结合使用, 例如 POP3。这样, 就能够对发送者发送给接收者的电子邮件进行排队, 接收者才能够检索新邮件。

接下来, 我们将讲述一下 POP3 协议。

POP3 (Post Office Protocol Version 3) 提供了一种对邮件信息进行排队的标准机制, 这样接收者才能够检索新邮件。

同 SMTP 一样, POP3 服务器也运行在 TCP/IP 之上, 并且它的默认端口为 110。也就是说, POP3 服务器在默认端口号 110 上监听。

POP3 是一种存储转发类型的信息传递服务, 当客户请求时, 它会把邮件转发给客户并且从 POP 服务器队列中删除。

POP3 客户利用用户名和口令使用基于口令的认证方法向 POP3 服务器认证。而这种认证是在一种加密的会话基础之上进行的, 因此这种认证是安全可靠的。认证完之后, POP3 客户发出的一系列命令被发送给 POP3 服务器。

在介绍完上述的基本概念之后, 我们将介绍 JavaMail 的体系结构。

在 JavaMail 系统的体系结构中, 与一个邮件服务器的会话是由 Session 对象封装的。Session 对象可以用于获取信息 Store 服务和信息 Transport 服务的句柄。Store 服务和 Transport 服务都扩展了一个一般的 Service 对象。邮件 Store 可以在一个或多个邮件 Folder 中存储信息。使用基于电子邮件信息属性的复杂的搜索标准, 同时利用层次性的 Searchterm 对象可以在一个文件夹中进行搜索信息。

电子邮件消息是由 Message 对象封装的, 同时通过使用一个具体的 MimeMessage 对象在 Internet 上封装电子邮件消息。Multipart 对象使用多个 Bodypart 元素封装消息。

JavaMail 也为邮件系统事件处理定义了一个事件模型。邮件系统事件是作为特殊的邮件事件对象实现的。对每一个邮件事件类型也定义了特殊的邮件事件监听器。

所有的这些邮件系统接口功能是在 JavaMail 所提供的四个包中实现的:

- **JavaX.Mail:** 定义了基本的 JavaMail 实用工具和邮件消息系统抽象。
- **JavaX.Mail.Internet:** 为使用 MIME 消息的 Internet 邮件系统的 JavaMail 抽象定义了一个具体的实现。
- **JavaX.Mail.Event:** 定义了邮件系统事件和监听器。
- **JavaX.Mail.Search:** 定义了一个层次性的搜索项, 当在邮件文件夹中搜索消息时使用他们表达的搜索标准。

## 9.4 JSP 在发信中的应用

下面讲述一下如何在一个 JSP 文件中发送 Mail, 下面的例子中需要导入类 sun.net.smtp.SmtpClient 和 java.io.\*。

用户输入信息的页面如图 9-32 所示, 其文件为 compose.htm。

代码如例程 9-24 所示。

例程 9-24

```
<html>
<head>
<title>在 JSP 中发送邮件</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb239">
</head>
<body bgcolor="#FFFFFF">
<div align="center">
<p><b>如何在 JSP 中发送邮件</b></p>
<form method="post" action="compose.jsp">
  <table width="70%" border="1">
    <tr>
      <td bgcolor="#CCCCFF" width="31%">
        <div align="right"><b><font color="#FF0033">收件人地址: </font></b></div>
      </td>
      <td bgcolor="#FFCCCC" width="69%">
        <input type="text" name="recipients" size="40">
      </td>
    </tr>
    <tr>
      <td bgcolor="#CCCCFF" width="31%">
        <div align="right"><b><font color="#FF0033">主题: </font></b></div>
      </td>
      <td bgcolor="#FFCCCC" width="69%">
        <input type="text" name="subject" size="40">
      </td>
    </tr>
    <tr>
      <td bgcolor="#CCCCFF" width="31%">
        <div align="right"><b><font color="#FF0033">内容: </font></b></div>
      </td>
      <td bgcolor="#FFCCCC" width="69%">
        <textarea name="content" cols="40" rows="6"></textarea>
      </td>
    </tr>
    <tr>
      <td bgcolor="#CCCCFF" width="31%">
```

```

        <div align="right"><b><font color="#FF0033">发信人地址: </font></b></div>
    </td>
    <td bgcolor="#FFCCCC" width="69%">
        <input type="text" name="addresser" size="40">
    </td>
</tr>
<tr>
    <td bgcolor="#CCCCFF" width="31%">
        <div align="right"><b><font color="#FF0033">邮件服务器: </font></b></div>
    </td>
    <td bgcolor="#FFCCCC" width="69%">
        <input type="text" name="host" size="40">
    </td>
</tr>
<tr bgcolor="#FF9966">
    <td colspan="2">
        <div align="center">
            <input type="submit" name="Submit" value="发 邮 件">
            <input type="reset" name="Submit2" value="重 写">
        </div>
    </td>
</tr>
</table>
</form>
<p>&nbsp;&nbsp;&nbsp;</p>
</div>
</body>
</html>

```

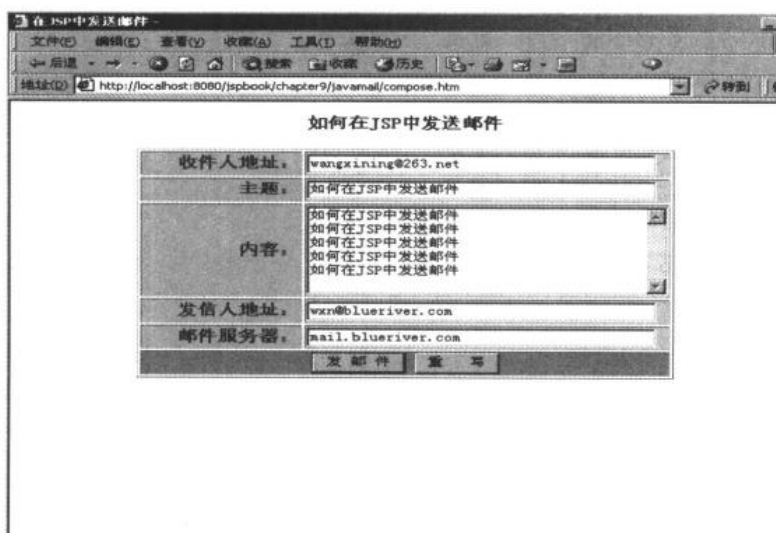


图 9-32 用户输入信息界面



填写完信息后（如上图所示），单击【发邮件】按钮，发送邮件。

把以下代码存为 `compose.jsp`，然后运行，即可发送邮件，发送后的页面如图 9-33 所示。

我们定义了以下变量：

- 变量 `addresser` 为发信人的地址
- 变量 `recipients` 为收信人的地址
- 变量 `subject` 为信件的标题
- 变量 `content` 为信件的内容
- 变量 `mailserver` 为发信的 SMTP 服务器地址

`compose.jsp` 文件的源码如例程 9-25 所示。

例程 9-25

```
<%@ page language="java" import="sun.net.smtp.SmtpClient,java.io.*" %>
<%@ page info="send mail in a jsp page"%>
<%
String addresser,recipients,subject,content,mailserver;
addresser=request.getParameter("addresser");
recipients=request.getParameter("recipients");
subject=request.getParameter("subject");
content=request.getParameter("content");
mailserver=request.getParameter("mailserver");
try
{
    SmtpClient host=new SmtpClient(mailserver);
    host.from(addresser);
    host.to(recipients);
    PrintStream MailMessage=host.startMessage();
    MailMessage.println("收件人地址:"+recipients);
    MailMessage.println("主题:"+subject);
    MailMessage.println("内容: "+content);
    MailMessage.println("发信人地址: "+addresser);
    MailMessage.println("邮件服务器: "+mailserver);
    host.closeServer();
    out.println("成功发送");
}
catch(IOException e)
{
    out.println(e.getMessage());
}
%>
```

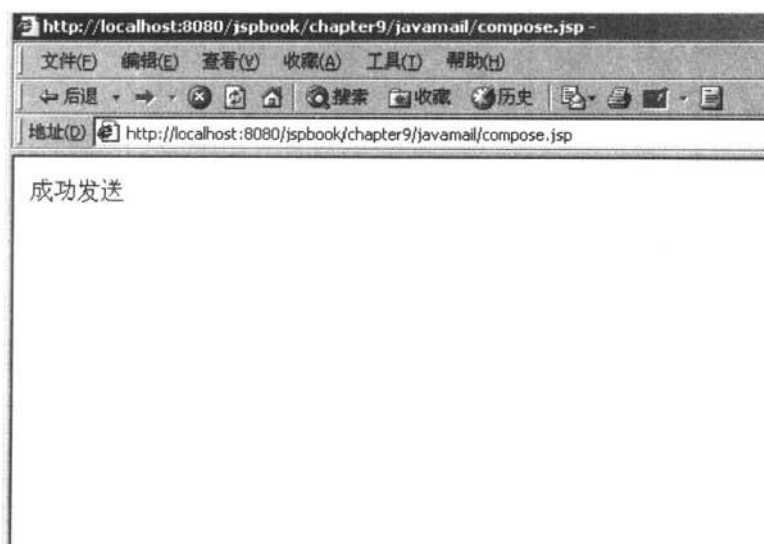


图 9-33 发送邮件后的界面



# 开发专家之 Sun ONE

## 第四篇 在 JSP 开发中使用数据库

在本篇中，将讲述在 JSP 开发中是如何使用数据库的。

本篇分为三章，分别讲述了 JSP 开发的数据库工作原理、在 JSP 和 Servlet 中是如何使用 JDBC 技术的，以及数据库的连接池技术。

在第 10 章中，讲述了 JSP 开发的数据库工作原理。首先讲述了目前最为流行的数据库及其工作原理，接着则介绍了 SQL Server 语句。在第二节中则是介绍了 JDBC 技术的工作原理。第三节则介绍了 JDBC 技术的四种类型的驱动，包括 JDBC-ODBC Bridge、JDBC Native Bridge、JDBC Net Bridge、Pure Java JDBC Driver。在第四节中讲述了 JDBC 接口 (Driver、DriverManager、Connection、Statement、ResultSet) 及其应用。最后一节则讲述了一个 JDBC 实例。

在第 11 章中，讲述了 JSP 与 Servlet 技术在实际开发中是如何使用 JDBC 技术的。在第 12 章中，讲述了数据库的连接池技术。

通过对本篇的学习，将会掌握如何在 JSP 开发中使用数据库。



# 第 10 章 JSP 开发中的数据库

## 工作原理

在本章中，将讲述 JSP 开发中的数据库工作原理。首先讲述了目前最为流行的数据库及其工作原理，接着则是介绍了 SQL 语句。在第二节中则是介绍了 JDBC 技术的工作原理。在第三节中则是介绍了 JDBC 技术的四种类型的驱动，包括 JDBC-ODBC Bridge、JDBC Native Bridge、JDBC Net Bridge、Pure Java JDBC Driver。在第四节中讲述了 JDBC 接口（Driver、DriverManager、Connection、Statement、ResultSet）及其应用。最后一节则是讲述了一个 JDBC 实例。

通过对本章的学习，将会掌握数据库在 JSP 开发中的应用。

### 10.1 数据库及 SQL 语句

数据库管理系统（DBMS）是一个软件系统，它具有存储、检索和修改数据的功能。数据库管理系统具有四个发展阶段：层次型、网状型、关系型和关系对象型。

目前，应用比较多的数据库有 Oracle、Sybase、Informix、Microsoft SQL Server 7.0/2000、DB2、MySQL 等。

Oracle 是目前国际上最为流行的关系型对象数据库。Oracle 已经成为全世界大、中型管理信息系统首选的数据库产品。Oracle 能够很好的支持 Java，Oracle 公司和 Sun 公司一并成为了 Microsoft 公司的两个竞争对手。笔者对 Oracle 产品也情有独钟。Oracle+Java 成为了一道亮丽的风景线。

Microsoft SQL Server 7.0/2000 是 Microsoft 公司推出的大型数据库系统，简单易用，而且能够和 Microsoft 公司的其他产品很好地结合。它的编程接口非常丰富、易用，可以很容易地用组件访问数据库。

另外，不得不提一下 Microsoft 公司的 Access。考虑到绝大多数的读者都用到 Access，而且 Access 也非常适合作为学习使用，所以本书的例子就是采用 Access。作为一个桌面数据库系统，Access 虽然不能用作企业使用，但完全可以在 PC 上供学习使用。

#### 10.1.1 建表、修改和删除表的语句

SQL 语言中的 CREATE TABLE 语句用来建立数据库的表。其基本语法为：

```
CREATE TABLE tableName(  
    column1 datatype [column_constraint],
```



column2 datatype [column\_constraint],

.....  
)

创建表，必须定义表名、列名、列的类型和列的宽度，一个表最多可以包含 2000 列。

### 10.1.2 查询语句

在 SQL 语句中，Select 语句应该是最为常用的语句。它的功能是实现数据库查询。查询语句的基本语法是：

SELECT ....	-----查询内容
FROM .....	-----表名
[WHERE ....]	-----条件
[GROUP BY .....	-----分组内容
[HAVING ....]	-----组内条件
[ORDER BY ...]	-----排序内容

### 10.1.3 插入、更新和删除语句

#### 1. 数据的插入

在 SQL 语句中，INSERT 语句是用来向表中添加记录的。INSERT 语句的基本语法是：

```
INSERT INTO tableName(column1, column2,.....)
values(values1, values2,.....)
```

#### 2. 数据的更新

要修改表中已经存在的一条或多条纪录，则应该使用 UPDATE 语句。UPDATE 语句可以使用 WHERE 语句来进行选择更新特定的纪录。

UPDATE 语句的基本语法是：

```
UPDATE tableName SET column1=values1, column2=values2,... [WHERE ...];
```

#### 3. 数据的删除

要删除表中已经存在的一条或多条纪录，则应该使用 DELETE 语句。DELETE 语句可以使用 WHERE 语句来进行选择删除特定的纪录。

DELETE 语句的基本语法是：

```
DELETE FROM tableName [ WHERE ...]
```

### 10.1.4 条件子句

#### 1. WHERE 语句

使用 WHERE 语句可以选择满足条件的特定的纪录。上面的几个例子中已经多次使用了 WHERE 语句。

在使用 WHERE 语句时，应注意：

若列的数据类型为数字型，则不需要用引号。

如：

```
delete from TOPIC where id=12
```

若列的数据类型为字符型，则需要用单引号把字符串括起来。

如：

```
update topic set author='author',email='aemail',content='content' where title='title';
```

- **IN 和 NOT IN**

选择列值与值列表中某一个值相等的相关行信息。相反，NOT IN 则选择那些不在列表中的纪录。

- **BETWEEN ... AND 和 NOT BETWEEN ... AND**

选择列值在某个范围的纪录相反地，NOT BETWEEN 选择列值不在该范围的纪录。

- **LIKE 和 NOT LIKE**

用于查找字符串的匹配。通配符“%”匹配任意长度的字符串，而“\_”只匹配一个字符。

- **IS NULL 和 IS NOT NULL**

用于查找列值为空值或非空值的纪录。



(1) 空值 (NULL) 不等同于零值。

(2) 零是数字，而空值 (NULL) 不是数字。

(3) 空值 (NULL) 不能像零那样进行算术运算。

(4) 空值 (NULL) 表示未知的、不存在的或不可用的数据。

- **逻辑运算 AND 和 OR**

逻辑运算 AND:

选择列值同时满足多个条件的纪录。

逻辑运算 OR:

选择列值满足其中任意一个条件的纪录。

- **ORDER BY 语句**

ORDER BY 语句用来确定纪录显示的先后顺序。

排序分为升序和降序两种，默认的情况是升序 (ASC)；如果在 ORDER BY 语句后加上“DESC”，那么显示的顺序为降序。

### 10.1.5 Oracle 的函数

Oracle 提供了大量的函数，可以大大地提高 SQL 语句的处理能力。这些函数通常用来修改数值、拼凑数值、改变数据格式等。

根据函数值的数据类型可划分为数值函数、字符函数、日期函数、转换函数以及其他函数。

根据函数对记录信息处理后返回的结果划分，可分为个体函数、聚组函数。

函数的一般格式为：

FUNC(Param1, Param2, ....., Paramn)

其中，FUNC 为函数名，Param1, Param2, ....., Paramn 为函数的参数。

## 10.2 JDBC 技术工作原理

JDBC 是 Java 数据库连接 (Java Data Base Connectivity) 技术的简称, 是为各种常用数据库提供无缝连接的技术。JDBC 技术是由 Sun 公司的 JavaSoft 公司制定的, JDBC 定义了 Java 语言同 SQL 数据之间的程序设计接口。JavaSoft 公司开发了 JDBC API, JDBC API 是一个标准统一的 SQL 数据存取接口。

JDBC API 为 Java 程序提供了一个统一无缝地操作各种数据库的接口, 程序员编程时, 可以不关心它所要操作的数据库是哪个厂家的产品, 从而提高了软件的通用性。JDBC 现在可以连接的数据库包括: Xbase、Oracle、Sybase、Access 以及 Paradox 等。

JDBC 在 Web 和 Internet 应用程序中的作用和 ODBC 在 Windows 系列平台应用程序中的作用类似。ODBC (OpenData Base Connectivity), 称为开放式数据库互联技术, 是由 Microsoft 公司倡导并得到业界普遍响应的一门数据库连接技术, 如果读者有使用 ODBC 编程的经验, 就会发现 JDBC 与 ODBC 很类似。

Java 语言具有健壮、安全、易于使用和易于从网络上下载等特性, 是编写数据库应用程序的最佳语言。Java 是“一次编写, 到处运行”的语言, 即 Java 程序不经改变即可部署到任何运行 Java 虚拟机的计算机结构和操作系统。对于大公司, 有一个公用开发平台好处很大, 编程人员可以不必再为大公司中的各个平台分别编程。Java 也很吸引第三方开发者, 单个 Java 程序即可满足大公司客户的需求。

建立公司系统中基于 Java 的应用程序和小程序的动力是巨大的。公司对于将结构和操作系统相关模型变成针对网络的模型相当有兴趣。Java 成为了节约资源成本的象征。

对于开发人员来说, Java 代表巨大的市场机会。大中型企业很少不用数据库进行业务工作的, 大部分公司将数据库应用到业务的各个方面, 从人事资源到前线客户销售。

JDBC 扩展了 Java 语言的功能。使用 JDBC, 向各种关系数据库发送 SQL 语句成为很容易的一件事。将 Java 和 JDBC 结合起来, 将会使程序经过一次编译即可在其他任何平台上运行。

为了建立独立于平台的应用程序和基于 Web 的小程序, 开发人员应考虑用 Java 开发前台连接的办法。从外部看, 第三方软件开发者通过专用办法, 通过本地方法集成客户机方案或通过建立第三层和新协议来满足这种需求。

JavaSoft 是 Sun 公司负责开发 Java 产品的业务单位, 和数据库及数据库工具厂家一起建立独立于 DBMS 的机制, 使开发人员不必考虑所用的特定数据库而编写客户机应用程序。产生的 JDBC API 第一版是核心 JDK 2 的一部分。

JDBC 向应用程序开发者提供了独立于数据库的统一的 API。这个 API 提供了编写的标准和考虑所有不同应用程序设计标准。其奥秘是一组由驱动程序实现的 Java 接口。驱动程序负责标准 JDBC 调用向支持的数据库所要的具体调用转变。

应用程序编写一次并移植到各种驱动程序上。应用程序不变, 驱动程序则各不相同。驱动程序可以用于开发多层数据库设计的中间层, 也称中间件 (middleware)。

除了向开发者提供统一的独立于 DBMS 的框架外, JDBC 还提供了让开发者保持数据

库厂家提供的特定功能的办法。JDBC 驱动程序必须支持 ANSI AQL-2 项目层，但 JDBC 允许开发者直接将查询字符串传递到连接的驱动程序。

JDBC 不是 Mincrosoft 的 ODBC(开放式数据库连接)规范派生的，JDBC 完全是用 Java 编写的，而 ODBC 是个 C 接口。但是，JDBC 和 ODBC 都是基于 X 或开放 SQL 命令层接口(CLI)，相同的概念性基础使 API 工作进展更快，使 API 的接受更加容易。JavaSoft 提供了将 JDBC 变成 ODBC 的 JDBC-ODBC 桥。这个用本地方法完成的版本很小很有效。

一般来说，JDBC API 中有两层接口：应用程序层，开发人员用 API 通过 SQL 调用数据库和取得结果；驱动程序层，处理与具体驱动程序版本的所有通信。

每个 JDBC 应用程序（或小程序）至少要有有一个 JDBC 驱动程序，每个驱动程序是针对一种 DBMS 的。但驱动程序不必直接连到数据库。

迄今为止，通过 Java 访问数据库的惟一方法就是利用 Java 中的流调和访问 Common Gateway Interface (CGI，公用网关接口) 程序。通过 Java 调用 CGI 脚本其实是执行一个访问数据库并返回结果的独立程序。

使用这种方法速度很慢，而且会在应用程序中引入更多的错误。这是由于利用两种不同的开发语言和开发程序，需要掌握两种不同的技术。使用 JDBC，用户只要了解 Java 语言即可，而使用 CGI，用户必须同时使用 Java 和另一种编程语言。

使用 JDBC 的另一个原因是它的速度比 CGI 方法快。使用 CGI 方法通常要求计算机执行另一个独立的程序。这个独立的程序访问数据库、处理数据，并将结果返回给调用程序。这就需要多级处理，因而增加了等待时间和出错概率。

调用 CGI 脚本通常是通过 Web 服务器执行一个新的脚本，而执行 JDBC 的数据库命令只需要将 SQL 命令发送给数据库的服务器，这就大大缩短了执行 SQL 语句的时间。CGI 脚本必须独立地连接数据库，处理执行结构，而 JDBC 的解决方案使应用程序直接与数据库相连，执行各种操作。

JDBC API 用于连接 Java 应用程序与各种关系数据库。这使得人们在建立客户/服务器应用程序时，通常把 Java 作为编程语言，把任何一种浏览器作为应用程序的友好界面，把 Internet 或 Intranet 作为网络主干，把有关的数据库作为数据库后端。以下是使用 JDBC 的优缺点。

优点如下：

- JDBC API 与 ODBC 十分相似，有利于用户理解。
- JDBC 使得编程人员从复杂的驱动器调用命令和函数中解脱出来，可以致力于应用程序中的关键地方。
- JDBC 支持不同的关系数据库，使得程序的可移植性大大加强。
- 用户可以使用 JDBC-ODBC 桥驱动器将 JDBC 函数调用转换为 ODBC。
- JDBC API 是面向对象的，可以让用户把常用的方法封装为一个类，以备后用。

缺点如下：

- 使用 JDBC，访问数据记录的速度会受到一定程度的影响。
- JDBC 结构中包含了不同厂家的产品，这就给更改数据源带来很大的麻烦。

JDBC 有一个非常独特的动态连接结构，它使得系统模块化。使用 JDBC 来完成对数据库的访问包括以下四个主要组件：Java 的应用程序、JDBC 驱动器管理器、驱动器和数

据源。

用 JDBC 来实现访问数据库记录可以采用下面的几个步骤：

#### 步骤

- (1) 通过驱动器管理器获取连接接口。
- (2) 获得 Statement 或它的子类。
- (3) 限制 Statement 中的参数。
- (4) 执行 Statement。
- (5) 查看返回的行数是否超出范围。
- (6) 关闭 Statement。
- (7) 处理其他的 Statement
- (8) 关闭连接接口。

## 10.3 JDBC 四种类型的驱动

### 10.3.1 JDBC-ODBC Bridge

在 JDBC 刚刚产生时，JDBC-ODBC 桥是非常有用的。通过 JDBC-ODBC 桥，开发者可以使用 JDBC 来访问一个 ODBC 数据源。

JDBC-ODBC 桥驱动程序为 Java 应用程序提供了一种把 JDBC 调用映射为 ODBC 调用的方法。因此，需要在客户端机器上安装一个 ODBC 驱动。

使用 JDBC-ODBC 桥的驱动器，通常只运行在 Microsoft Windows 系统，因此 JDBC 平台无关性的好处就不再存在。况且，ODBC 驱动器还需要客户端的管理。

由于 ODBC 是目前流行的 DBMS 接口标准，已经有许多可用的 ODBC 驱动程序与大量数据库交互，而且由于 JDBC 设计时考虑了 ODBC，Sun 公司在 JDK1.1 和 J2SE 中提供了 JDBC-ODBC 驱动程序实现。

通常不推荐使用这种桥驱动程序，但它可以减少开发人员进行企业开发的麻烦。

### 10.3.2 JDBC Native Bridge

JDBC-Native 桥提供了一个建筑在本地数据库驱动上的 JDBC 接口。JDBC 驱动将标准的 JDBC 调用转变为对数据库 API 的本地调用，该类型的驱动程序是本地部分 Java 技术性能的本机 API 驱动程序。

使用该类型的驱动也将会失去 JDBC 平台无关性的好处，并且需要安装客户端的本地代码。

现在大多数的数据库厂商都在其数据库产品中提供该桥驱动程序，这些驱动程序大多数都提供比使用 JDBC-ODBC 桥驱动程序更好的性能。



### 10.3.3 JDBC-Network Bridge

JDBC-network 桥不需要客户端的数据库驱动, 而是使用网络-服务器中层来访问一个数据库。该类型的驱动程序是网络协议完全 Java 技术性能的驱动程序, 它为 Java 应用程序提供了一种进行 JDBC 调用的机制。

使用该类型的驱动程序是平台无关的, 并且不需要客户端的安装和管理, 因此很适合用做 Internet 的应用。

### 10.3.4 Pure Java JDBC Drive

纯 Java 驱动运行在客户端, 并且直接访问数据库, 因此运行这个模式要使用一个两层的体系。要在一个 n 层的体系中使用该类型的驱动, 可以通过一个包含有数据访问代码的 EJB, 并且让该 EJB 为它的客户提供与数据库无关的服务。

该类型的驱动程序是本地协议完全 Java 技术性能的驱动程序, 它为 Java 应用程序提供了一种进行 JDBC 调用的机制。

## 10.4 JDBC 接口

### 10.4.1 Driver

Driver 的定义是非常简单的, Driver 要给出要用到的 class name。

如果使用 JDBC-ODBC Bridge Driver, 那么定义是这样的:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

而如果使用 JDBC Driver, 那么定义是这样的:

```
Class.forName("jdbc.driver_class_name");
```

在使用 Class.forName 之前, 应先导入 import 语句。

```
import java.sql.*;
```

// 连接 JDBC-ODBC 桥驱动

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

//连接 MySQL Jdbc 驱动

```
Class.forName("org.gjt.mm.mysql.Driver");
```

// 连接 Oracle Jdbc 驱动

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

### 10.4.2 DriverManager

要创建一个连接对象实例, 必须以数据库统一资源定位器作为参数去激发 DriverManager 的 getConnection()方法。而所谓的数据库统一资源定位器 (URL) 是指一个完全合格的数据库连接的名称, 该名称标识所要连接的数据库和数据库连接程序。如下



面形式的一个字符串表示：JDBC:PROTOCOL:NAME。

如 jdbc:oracle:thin:@local-host:1521:orcl。

其中：

JDBC 是指对所有的 JDBC 数据库 URL 的 URL 中所用的数据库驱动程序类型关键字；

PROTOCOL 是指用户要连接的数据库的类型；

NAME 提供建立连接时所需的数据库类型的额外信息。

DriverManager（驱动程序管理器）类是 JDBC（Java 数据库连接）的管理层，作用于用户和驱动程序之间。DriverManager 类跟踪可用的驱动程序，并在数据库和相应驱动程序之间建立连接。另外，DriverManager 类也处理诸如驱动程序登录时间限制及登录和跟踪消息的显示等事务。

当 DriverManager 激发 getConnection()方法时，DriverManager 类首先从它已加载的驱动程序池中找到一个可以接受该数据库 URL 的驱动程序，然后请求该驱动程序使用相关的数据库 URL 去连接到数据库中。于是，getConnection()方法建立了与数据库的连接。

JDBC 允许用户使用调用 DriverManager 的方法 getDriver、getDrivers 和 registerDriver 及 Driver 的方法 connect。

DriverManager 类包含一系列的驱动程序类，这些驱动程序类已通过调用方法 DriverManager.registerDriver 对自己进行了注册。所有的驱动程序类都必须包含有一个静态部分。这个静态部分创建该类的实例，然后在加载该实例的 DriverManager 类时进行注册。这样，用户在正常情况下将不会直接调用 DriverManager.registerDriver 方法；而是在加载驱动程序时由驱动程序自动调用。

加载 Driver 类，然后自动在 DriverManager 中注册的方式有两种：

#### 1. 通过调用方法 Class.forName。

这将显式地加载驱动程序类。由于这与外部设置无关，因此推荐使用这种加载驱动程序的方法。

如以下代码加载类 oracle.jdbc.driver.OracleDriver：

```
<%
    String sDBDriver="oracle.jdbc.driver.OracleDriver";
    try
    {
        Class.forName(sDBDriver);
    }
    catch(java.lang.ClassNotFoundException e)
    {
        System.err.println(e.getMessage());
    }
%>
```

如果将 oracle.jdbc.driver.OracleDriver 编写为加载时创建实例，并调用以该实例为参数的 DriverManager.registerDriver，则它位于 DriverManager 的驱动程序列表中，并可用来创建连接。

## 2. 通过将驱动程序添加到 Java.lang.System 的属性 jdbc.drivers 中

要隐式地加载一组驱动程序，只需要在 JDBC.Drivers 系统属性中指定由冒号（:）分隔的驱动程序类的名称即可。

在初始化 DriverManager 类时，它搜索系统属性 jdbc.drivers，如果用户已输入了一个或多个驱动程序，则 DriverManager 类将试图加载它们。

要注意所指定的驱动程序类的名称必须在用户的 ClassPath 中，例如：

```
Java -Djdbc.drivers= sun.jdbc.odbc.JdbcOdbcDriver:com.asuredtech.jdbc.JdbcDriver Prgrm
```

对 DriverManager 方法的第一次调用将自动加载这些驱动程序类。



加载驱动程序的第二种方法需要持久的预设环境。如果对这一点不能保证，则调用方法 Class.forName 显式地加载每个驱动程序就显得更为安全。这也是引入特定驱动程序的方法，因为一旦 DriverManager 类被初始化，它将不再检查 jdbc.drivers 属性列表。

在以上两种情况中，新加载的 Driver 类都要通过调用 DriverManager.registerDriver 类进行自我注册。如上所述，加载类时将自动执行这一过程。

由于安全方面的原因，JDBC 管理层将跟踪类加载器提供的驱动程序。这样，当 DriverManager 类打开连接时，它仅使用本地文件系统或使用发出连接请求且代码相同的类加载器提供的驱动程序。

加载 Driver 类并在 DriverManager 类中注册后，就可以用来与数据库建立连接。当调用 DriverManager.getConnection 方法发出连接请求时，DriverManager 将检查每个驱动程序，查看它是否可以建立连接。

要创建一个连接对象实例，就必须以数据库的 URL 作为参数去激发 DriverManager 的一个 getConnection() 方法。

当激发这个方法时，DriverManager 首先从它已经加载的驱动程序池中找到一个可以接受数据库 URL 的驱动程序，然后请求该驱动程序使用相关的数据库 URL 去连接到数据库上。然后，给激发 DriverManager.getConnection() 方法的对象返回一个连接对象。

有时可能有多个 JDBC 驱动程序可以与给定的 URL 连接。在与给定远程数据库连接时，可以使用 JDBC-ODBC 桥驱动程序、JDBC 到通用网络协议驱动程序或数据库厂商提供的驱动程序。在这种情况下测试驱动程序的顺序至关重要，因为 DriverManager 将使用它所找到的第一个可以成功连接到给定 URL 的驱动程序。

首先 DriverManager 试图按注册的顺序使用每个驱动程序（jdbc.drivers 中列出的驱动程序总是先注册）。它将跳过代码不可信任的驱动程序，除非加载它们的源与试图打开连接的代码的源相同。

它通过轮流在每个驱动程序上调用方法 Driver.connect，并向它们传递用户开始传递给方法 DriverManager.getConnection 的 URL 来对驱动程序进行测试，然后连接第一个认出该 URL 的驱动程序。

这种方法初看起来效率不高，但由于不可能同时加载数十个驱动程序，因此每次连接实际只需几个过程调用和字符串比较。

在 DriverManager 类上存在三种形式的 getConnection() 方法：

- getConnection( String url): 只是简单地给定数据库 URL, 然后尝试连接。
- getConnection(String url, String User, String Password): 给定数据库 URL、数据库的用户名、数据库的用户使用的密码, 然后尝试连接。
- getConnection(String url, java.util.Properties information): 给定数据库的 URL 以及一个属性集合作为参数, 然后尝试连接。

例程 10-1 的代码是通常情况下用驱动程序建立连接所需步骤的范例。

例程 10-1

```
String sDBDriver="oracle.jdbc.driver.OracleDriver";
try
{
    Class.forName(sDBDriver);
}
catch(java.lang.ClassNotFoundException e)
{
    System.err.println(e.getMessage());
}

String user="system";
String pwd="manager";
String sConnStr="jdbc:oracle:thin:@localhost:1521:orcl";
Connection conn=null;
ResultSet rs=null;
try
{
    conn=DriverManager.getConnection(sConnStr,user,pwd);
    Statement stmt=conn.createStatement();
    rs=stmt.executeQuery(sql);
}
catch(SQLException ex)
{
    System.err.println(ex.getMessage());
}
```

### 10.4.3 Connection

Connection 对象是代表与数据库的连接, 也就是在已经加载的 Driver 和数据库之间建立连接。您必须创建一个 Connection class 的实例, 其中包括您的数据库的信息。

连接过程包括所执行的 SQL 语句和在该连接上所返回的结果。一个应用程序可与单个数据库有一个或多个连接, 或者可与许多数据库有连接。

DriverManager 的 getConnection() 方法, 将建立在 JDBC URL 中定义的数据库的 connection 连接上:

```
Connection conn = DriverManager.getConnection(url, login, password);
```

根据不同的 Driver, URL 可能是不同的, 但常常有类似如下的格式:

```
jdbc:driver-id:database-id
```

```
jdbc:driver-id://host/database-id
```

例如, Oracle 提供了 2 种 JDBC drivers:

- JDBC Thin for Java applets and applications
- JDBC OCI for Java applications

按照所用的 JDBC 的不同, URL 也有所不同。

```
import java.sql.*;
// JDBC-ODBC 桥驱动
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection dbCon = DriverManager.getConnection(URL);
// MySQL Jdbc 驱动
Class.forName("org.gjt.mm.mysql.Driver");
Connection dbCon = DriverManager.getConnection(jdbc:mysql:/// URL);
// Oracle Jdbc Thin 驱动
Class.forName("oracle.jdbc.driver.OracleDriver");
String url="jdbc:oracle:thin:@192.168.1.31:1521:member";
Connection dbCon = DriverManager.getConnection(url, "system", "manager");
// Oracle Jdbc OCI 驱动
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection dbCon = DriverManager.getConnection("jdbc:oracle:thin:@ 192.168.1.31:1521:orcl", "system", "manager");
```

#### 10.4.4 Statement

Statement 对象用于将 SQL 语句发送到数据库中。实际上有三种 Statement 对象, 它们都作为在给定连接上执行 SQL 语句的容器: Statement、PreparedStatement (它从 Statement 继承而来) 和 CallableStatement (它从 PreparedStatement 继承而来)。

当打开一个数据库的连接后, Java 应用程序通常通过创建与执行一系列的 SQL 语句命令来实际使用该连接。这些 SQL 命令作为数据库的语句执行, 完成一些操作。

语句接口提供了可以被执行的基本的 SQL 语句, 作为提高性能的一项措施, PreparedStatement 对象提供了可以与查询信息一起预编译的一种语句类型。CallableStatement 对象是一种类型的 PreparedStatement, 它是用来封装数据库中存储过程的执行。

它们都专用于发送特定类型的 SQL 语句: Statement 对象用于执行不带参数的简单 SQL 语句; PreparedStatement 对象用于执行带或不带 IN 参数的预编译 SQL 语句; CallableStatement 对象用于执行对数据库已存储过程的调用。

建立了到特定数据库的连接之后, 就可用该连接发送 SQL 语句。Statement 对象用 Connection 的方法 createStatement 创建, 如下列代码所示:

```
String user="system";
String pwd="manager";
String sConnStr="jdbc:oracle:thin:@localhost:1521:orcl";
Connection conn=null;
ResultSet rs=null;
```

```
try
{
    conn=DriverManager.getConnection(sConnStr,user,pwd);
    Statement stmt=conn.createStatement();
    rs=stmt.executeQuery(sql);
}
catch(SQLException ex)
{
    System.err.println(ex.getMessage());
}
```

为了执行 Statement 对象，被发送到数据库的 SQL 语句将被作为参数提供给 Statement 的方法：

```
ResultSet rs = Statement.executeQuery("select * from member");
```

Statement 接口提供了三种执行 SQL 语句的方法：executeQuery、executeUpdate 和 execute。使用哪一个方法由 SQL 语句所产生的内容决定。

调用 Statement 对象，使用下列三种基本方法中的一种来执行一条实际的 SQL 命令。

#### 1. ResultSet executeQuery( String sql)

允许用户执行 SQL 语句查询并能够获得一个 ResultSet 对象。

用于产生单个结果集的语句，例如 SELECT 语句。

例如：

```
ResultSet RS= Statement.executeQuery("select * from member where name ='blueriver'");
```

#### 2. Int executeUpdate(String sql)

允许用户执行 SQL 插入、删除以及更新，然后获得被更新的行的数目。

用于执行 INSERT、UPDATE 或 DELETE 语句以及 SQL DDL（数据定义语言）语句。

返回值是一个整数，指示受影响的行数（即更新计数）。对于 CREATE TABLE 或 DROP TABLE 等不操作行的语句，executeUpdate 的返回值总为零。

例如：

```
String sqlinsert="insert into
member(logname,realname,password,email,gender,phone,problem,answer,
province,education,hobbies,selfintro)
Values('"+logname+"','"+realname+"','"+passwd+"','"+email+"','"+gender+"','"+phone+"','"+problem
+"','"+answer+"','"+province+"','"+education+"','"+hobby+"','"+selfintro+"')";
int num=statement.executeUpdate(sqlinsert);
```

#### 3. Boolean execute(String sql)

执行 SQL 语句调用最一般的方法，允许用户执行 SQL 数据定义语言命令，然后获得一个布尔值，它显示是否返回了 ResultSet 对象。

用于执行返回多个结果集、多个更新计数或两者组合的语句。

例如：

```
String sql=" select * from member";
boolean result=statement.execute(sql);
```

执行语句的所有方法都将关闭所调用的 Statement 对象的当前打开结果集。这意味着



在重新执行 Statement 对象之前，需要完成对当前 ResultSet 对象的处理。

继承了 Statement 接口中所有方法的 PreparedStatement 接口都有自己的 executeQuery、executeUpdate 和 execute 方法。

Statement 对象本身不包含 SQL 语句，因而必须给 Statement.execute 方法提供 SQL 语句作为参数。PreparedStatement 对象并不将 SQL 语句作为参数提供给这些方法，因为它们已经包含预编译 SQL 语句。CallableStatement 对象继承这些方法的 PreparedStatement 形式。对于这些方法的 PreparedStatement 或 CallableStatement 版本，使用查询参数将抛出 SQL Exception。

当连接处于自动提交模式时，其中所执行的语句在完成时将自动提交或还原。语句在已执行且所有结果返回时，即认为已完成。对于返回一个结果集的 executeQuery 方法，在检索完 ResultSet 对象的所有行时该语句完成。对于方法 executeUpdate，当它执行时语句即完成。但在少数调用方法 execute 的情况中，在检索所有结果集或它生成的更新计数之后语句才完成。

有些 DBMS 将已存储过程中的每条语句视为独立的语句；而另外一些则将整个过程视为一个复合语句。在启用自动提交时，这种差别就变得非常重要，因为它影响什么时候调用 commit 方法。在前一种情况中，每条语句单独提交；在后一种情况中，所有语句同时提交。

Statement 对象将由 Java 垃圾收集程序自动关闭。而作为一种好的编程风格，应在不需要 Statement 对象时显式地关闭它们。这将立即释放 DBMS 资源，有助于避免潜在的内存问题。

Statement 可包含使用 SQL 转义语法的 SQL 语句。转义语法告诉驱动程序其中的代码应该以不同的方式处理。驱动程序将扫描任何转义语法，并将它转换成特定数据库可理解的代码。这使得转义语法与 DBMS 无关，并允许程序员使用在没有转义语法时不可用的功能。

execute 方法应该仅在语句能返回多个 ResultSet 对象、多个更新计数或 ResultSet 对象与更新计数的组合时使用。当执行某个已存储过程或动态执行未知 SQL 字符串（即应用程序程序员在编译时未知）时，有可能出现多个结果的情况，尽管这种情况很少见。例如，用户可能执行一个已存储过程，并且该已存储过程可执行更新，然后执行选择，再进行更新，再进行选择等。通常使用已存储过程的人应知道它所返回的内容。

因为方法 execute 的处理非常规情况，所以获取其结果需要一些特殊处理并不足为怪。例如，假定已知某个过程返回两个结果集，则在使用方法 execute 执行该过程后，必须调用方法 getResultSet 获得第一个结果集，然后调用适当的 getXXX 方法获取其中的值。要获得第二个结果集，需要先调用 getMoreResults 方法，然后再调用 getResultSet 方法。如果已知某个过程返回两个更新计数，则首先调用方法 getUpdateCount，然后调用 getMoreResults，并再次调用 getUpdateCount。

对于不知道返回内容的，则情况更为复杂。如果结果是 ResultSet 对象，则方法 execute 返回 true；如果结果是 Java int，则返回 false。如果返回 int，则意味着结果是更新计数或执行的语句是 DDL 命令。在调用方法 execute 之后要做的第一件事情是调用 getResultSet 或 getUpdateCount 方法。调用方法 getResultSet 可以获得两个或多个 ResultSet 对象中的第



一个对象，或调用方法 `getUpdateCount` 可以获得两个或多个更新计数中第一个更新计数的内容。

当 SQL 语句的结果不是结果集时，则方法 `getResultSet` 将返回 `null`。这可能意味着结果是一个更新计数或没有其他结果。在这种情况下，判断 `null` 真正含义的惟一方法是调用方法 `getUpdateCount`，它将返回一个整数。这个整数为调用语句所影响的行数，如果为 `-1` 则表示结果是结果集或没有结果。如果方法 `getResultSet` 已返回 `null`（表示结果不是 `ResultSet` 对象），则返回值 `-1` 表示没有其他结果。

如果已经调用方法 `getResultSet` 并处理了它返回的 `ResultSet` 对象，则有必要调用方法 `getMoreResults` 以确定是否有其他结果集或更新计数。如果 `getMoreResults` 返回 `true`，则需要再次调用 `getResultSet` 来检索下一个结果集。

综上所述，如果 `getResultSet` 返回 `null`，则需要调用 `getUpdateCount` 来检查 `null` 是表示结果为更新计数还是表示没有其他结果。

当 `getMoreResults` 返回 `false` 时，它表示该 SQL 语句返回一个更新计数或没有其他结果。因此需要调用方法 `getUpdateCount` 来检查它是哪一种情况。

#### 10.4.5 ResultSet

执行 SQL 语句，则 `ResultSet` 会包含符合 SQL 语句中条件的所有行。通过 `get` 方法，可以访问当前行中的不同列，`get` 方法也提供了对这些行中数据的访问。

`ResultSet.next()` 方法用于移动到 `ResultSet` 中的下一行，使下一行成为当前行。

结果集一般是一个表，其中有查询所返回的列标题及相应的值。

每调用一次 `next` 方法，光标向下移动一行。最初它位于第一行之前，因此第一次调用 `next` 将把光标置于第一行上，使它成为当前行。随着每次调用 `next` 导致光标向下移动，按照从上至下的次序获取 `ResultSet` 行。

方法 `getXXX` 提供了获取当前行中某列值的途径。在每一行内，可按任何次序获取列值。但为了保证可移植性，应该从左至右获取列值，并且一次性地读取列值。

列名或列号可用于标识要从中获取数据的列。例如，如果 `ResultSet` 对象 `rs` 的第二列名为“`menu_name`”，并将值存储为字符串，则下列任一代码将获取存储在该列中的值：

```
String s = rs.getString("menu_name");  
String s = rs.getString(2);
```



列是从左至右编号的，并且从列 1 开始。同时，用做 `getXXX` 方法输入的列名不区分大小写。

提供使用列名这个选项的目的，是为了让在查询中指定列名的用户可使用相同的名字作为 `getXXX` 方法的参数。如果 `select` 语句未指定列名，则应该使用列号。这些情况下，用户将无法确切知道列名。

有些情况下，SQL 查询返回的结果集中可能有多个列具有相同的名字。如果列名用作 `getXXX` 方法的参数，则 `getXXX` 将返回第一个匹配列名的值。因而，如果多个列具有相同的名字，则需要使用列索引来确保检索了正确的列值。这时，使用列号效率要稍微

高一些。

如果调用一个 `ResultSet` 对象上的 `getMetadata()` 方法, 将获得 `Java.sql.ResultSetMetadata` 对象的一个句柄, 可以用它来提供所返回的 `ResultSet` 对象的动态信息。

对 `ResultSetMetadata` 对象, 可以进行下列一些主要形式的调用:

- `int getColumnCount()`: 返回与本 `ResultSet` 对象相关联的列的数目
- `String getColumnName(int column)`: 返回一个列的名字
- `String getColumnClassName(int column)`: 返回一个列的类名字
- `String getColumnLabel(int column)`: 返回为显示的目的而推荐的列标签
- `int getColumnDisplaySize(int column)`: 返回该列的最大字符宽度
- `int getColumnType(int column)`: 返回某一列的 `Java.sql.Types` 常量的值
- `String getColumnName(int column)`: 返回与数据库相关的该列的类型名字
- `String getSchemaName(int column)`: 返回与一个列相关联的表的模式名
- `String getCatalogName(int column)`: 返回一个列的表目录名称
- `String getTableName(int column)`: 返回与一个列相联系的表的名字
- `String isCaseSensitive(int column)`: 指明该列是否区分大小写
- `boolean isReadOnly(int column)`: 指明该列是否为只读
- `boolean isWritable(int column)`: 指明是否可以对列进行写操作
- `boolean isDefinitelyWritable(int column)`: 指明对此列的一个写操作是否绝对可以进行
- `int isNullable(int column)`: 指明该列是否可以空值
- `boolean isSearchable(int column)`: 指明该列是否可以被用于 `where` 子句中

对于 `getXXX` 方法, JDBC 驱动程序试图将基本数据转换成指定 `Java` 类型, 然后返回适合的 `Java` 值。

例如, 如果 `getXXX` 方法为 `getFloat`, 而基本数据库中数据类型为 `int`, 则 JDBC 驱动程序将把 `int` 转换成 `Java Float`。 `getFloat` 的返回值将为 `Java Float` 对象。

`ResultSet` 可以获取任意大的 `LongVarbinary` 或 `LongVarchar` 数据。方法 `getBytes` 和 `getString` 将数据返回为大的块 (最大为 `Statement.getMaxFieldSize` 的返回值)。但是, 以较小的固定块获取非常大的数据可能会更方便, 而这可通过让 `ResultSet` 类返回 `Java.io.InputStream` 来完成。从该流中可分块读取数据。其中需要注意的识必须立即访问这些流, 因为在下一次对 `ResultSet` 调用 `getXXX` 方法时它们将自动关闭。

JDBC API 具有三个获取流的方法, 分别具有不同的返回值:

- `getBinaryStream` 返回只提供数据库原字节而不进行任何转换的流
- `getAsciiStream` 返回提供单字节 ASCII 字符的流
- `getUnicodeStream` 返回提供双字节 Unicode 字符的流

要确定给定结果的值是否是 JDBC Null, 必须先读取该列, 然后使用 `ResultSet.wasNull` 方法检查该次读取是否返回 JDBC Null。

当使用 `ResultSet.getXXX` 方法读取 JDBC NULL 时, 方法 `wasNull` 将返回下列值之一:

- `Java null` 值

对于返回 `Java` 对象的 `getXXX` 方法的值。例如 `getString`、`getBigDecimal`、`getBytes`、

getDate、getTime、getTimestamp、getAsciiStream、getUnicodeStream、getBinaryStream、getObject 等。

- 零值

对应于 getByte、getShort、getInt、getLong、getFloat 和 getDouble 方法。

- false 值

对应于 getBoolean 方法。

通常使用 executeQuery 语句返回单个 ResultSet，使用 executeUpdate 语句进行任何数据库修改，并返回更新行数。

但有些情况下，应用程序在执行语句之前不知道该语句是否返回结果集。此外，有些已存储过程可能返回几个不同的结果集或更新计数。

为了适应这些情况，JDBC 提供了一种机制，允许应用程序执行语句，然后处理由结果集和更新计数组成的任意集合。这种机制的原理是首先调用一个完全通用的 execute 方法，然后调用另外三个方法，getResultSet、getUpdateCount 和 getMoreResults。这些方法允许应用程序一次一个地研究语句结果，并确定给定结果是 ResultSet 还是更新计数。

用户不必关闭 ResultSet，当产生它的 Statement 关闭、重新执行或用于从多结果序列中获取下一个结果时，该 ResultSet 将被 Statement 自动关闭。

## 10.5 数据库连接 JDBC 实例

### 10.5.1 建立与 Oracle 的连接

建立一个 Oracle 的数据库。设计表名为 member，字段分别为 userId（字符型，长度为 50）、password（字符型，长度为 50）。

UserId 存储用户的 id，password 则用来存储用户的密码。如例程 10-2 所示。

例程 10-2

```
<%@ page info="database handler"%>
<%@ page import="Java.io.*"%>
<%@ page import="Java.util.*"%>
<%@ page import="Java.sql.*"%>
<%@ page import="Javax.servlet.*"%>
<%@ page import="Javax.servlet.http.*"%>
<html>
<body>
<%
//以 try 开始
try
{
    Connection con;
    Statement stmt;
```

```
ResultSet rs;
//加载驱动程序, 下面的代码为加载 JDBC-ODBC 驱动程序
//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Class.forName("oracle.jdbc.driver.OracleDriver");
//用适当的驱动程序连接到数据库, member 是系统 dsn 名
String url="jdbc:oracle:thin:@localhost:1521:member";
//建立连接, 类似于 ASP 中的创建数据库联接
con=DriverManager.getConnection(url, "system", "manager");
out.println("dfd");
//创建一个 JDBC 声明
stmt = con.createStatement();
//增加新记录
stmt.executeQuery("INSERT INTO member (userId,password) VALUES ('1','2')");
//查询记录
rs = stmt.executeQuery("SELECT userId,password from member");
//输出查询结果
out.println("<table border=1 width=400>");
while (rs.next())
{
String col1 = rs.getString(1);
String col2 = rs.getString(2);
//打印所显示的数据
out.println("<tr><td>"+col1+"</td><td>"+col2+"</td></tr>");
}
out.println("</table>");
}
//如果加载时出错, 给出相应的错误信息
catch (Exception e) {
out.println(e);
}
%>
</body>
</html>
```

### 10.5.2 建立与 My SQL 的连接

建立一个 MySQL 的数据库。设计表名为 member, 字段分别为 UserId (字符型, 长度为 50)、password (字符型, 长度为 50)。

UserId 存储用户的 id, password 则用来存储用户的密码。如例程 10-3 所示。

例程 10-3

```
<%@ page info="database handler"%>
<%@ page import="Java.io.*"%>
<%@ page import="Java.util.*"%>
<%@ page import="Java.sql.*"%>
<%@ page import="Javax.servlet.*"%>
```

```
<%@ page import="javax.servlet.http.*"%>
<%@ page contentType="text/html; charset=8859_1" %>
<%
//声明
Java.sql.Connection sqlConn;//数据库连接对象
Java.sql.Statement sqlStmt;//语句对象
Java.sql.ResultSet sqlRst;//结果集对象
//登记 JDBC 驱动程序
Class.forName("org.gjt.mm.mysql.Driver").newInstance();
//连接数据库
sqlConn = Java.sql.DriverManager.getConnection("jdbc:mysql://localhost/test","test","test");
//创建语句对象
sqlStmt
sqlConn.createStatement(Java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE,Java.sql.ResultSet.CONCUR
_READ_ONLY);
//执行 SQL 语句
sqlRst = sqlStmt.executeQuery("SELECT userId,password from member");
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>Linux-JSP-JDBC-MySQL 测试 - Select</title>
</head>
<body>
<table border="1" cellspacing="0" cellpadding="0" align="center">
<tr>
<th>姓名</th>
<th>年龄</th>
</tr>
<%while(sqlRst.next()){ %>
<tr>
<td><%=sqlRst.getString(1)%></td>
<td><%=sqlRst.getLong(2)%></td>
</tr>
<%}%>
</table>
</body>
</html>
<%
//关闭结果集对象
sqlRst.close();
//关闭语句对象
sqlStmt.close();
//关闭数据库连接
sqlConn.close();
%>
```

## 10.5.3 实例

下面我们来看一个例子，该例子是一个用 Web 形式实现树状菜单的例子（功能非常像资源管理器，不过这只是一个 Web 形式而已）。

该例子是笔者在实际的项目开发过程中用过的，在这里笔者想通过这个例子来讲述用 JDBC 连接 Oracle 数据库的实际应用。

这个例子采用的数据库是 Oracle，表名以及其结构如下：

表名：MENU\_CONTENT

功能：保存菜单下上传文件的相关信息的纪录

表结构如表 10-1 所示。

表 10-1 数据库表结构说明

字段名称	数据类型	说 明
ID	自动编号	关键字
MENU_ID	数字	菜单 ID
UPLOAD_FILE_NAME	文本	上传的文件名
UPLOAD_TIME	时间	上传的时间
UPLOAD_MEMBER_NAME	文本	上传人名

表名：TREE\_MENU

功能：保存菜单的相关信息的纪录

表结构如表 10-2 所示。

表 10-2 TREE\_MENU 表说明

字段名称	数据类型	说 明
ID	自动编号	关键字
MENU_ID	数字	菜单 ID
MENU_NAME	文本	菜单名称
PARENT_MENU_ID	数字	父菜单 ID
FOLDER_OR_FILE	数字	文件夹还是文件
MENU_LOCATION	数字	菜单位置

该例子中的所有文件及其描述（所有的文件在 mis 目录下）如表 10-3 所示。

表 10-3 文件及功能描述

文件\文件夹	功 能 描 述
index.jsp	主页面（框架结构，分为左右两部分，左边的页面是 basetree.htm，右边的页面是 basefolder.htm）



(续表)

文件\文件夹	功能描述
main.jsp	右边的显示页面，当点击左边的菜单名称时，右边的页面将会动态显示相应的页面内容
Basetree.htm	左边的菜单页面
basefolder.htm	初始化的右边的信息页面
add.jsp	动态增加菜单
add_ok.jsp	动态增加菜单
del.jsp	动态删除菜单
del_ok.jsp	动态删除菜单
edit.jsp	动态编辑菜单
edit_ok.jsp	动态编辑菜单
upload.htm	上传页面
upload.jsp	上传页面
js\all.js	Javascript
Upload 文件夹	用来放置上传文件的文件夹

以下是每个文件的部分代码及相关的解释：

#### 1. index.jsp

运行效果如图 10-1 所示，源代码如例程 10-4 所示。

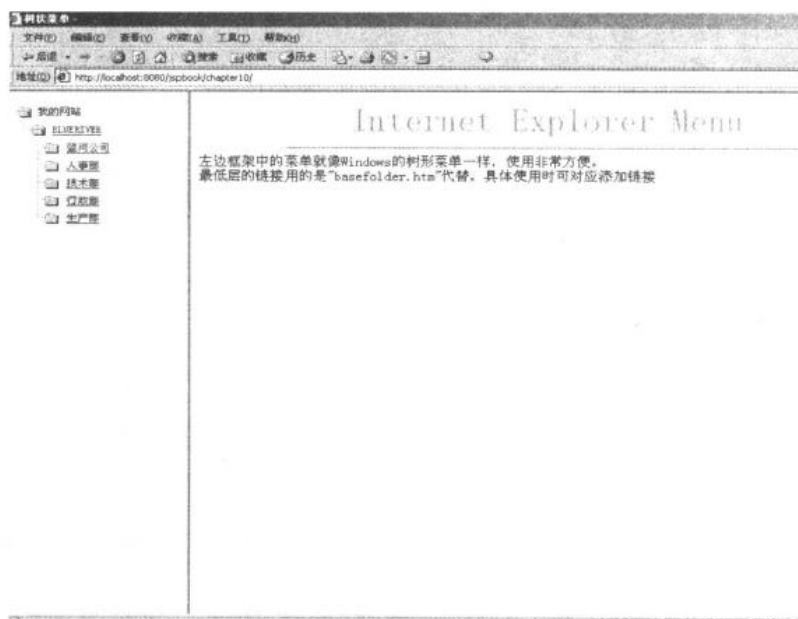


图 10-1 运行效果图

例程 10-4

```
<%@page contentType="text/html;charset=gb2312"%>
```

```
<%@page language="Java" import="java.sql.*"%>
<%
//设置 session, 提供用户登录接口
//这是跟用户系统的接口, 可随着不同的情况, 作以相应的改变
session.setAttribute("userName","blueriver");
String topMenu;
topMenu=(String)session.getAttribute("userName");
%>
<%
    String sDBDriver="oracle.jdbc.driver.OracleDriver";
    try
    {
        //登记 jdbc 驱动程序
        Class.forName(sDBDriver);
    }
    catch(java.lang.ClassNotFoundException e)
    {
        out.println(e.getMessage());
    }
%>
<%!
    public ResultSet executeQuery(String sql)
    {
        String user="system";
        String pwd="manager";
        String sConnStr="jdbc:oracle:thin:@localhost:1521:orcl";
        //数据库连接对象
        Connection conn=null;
        //语句对象
        ResultSet rs=null;
        //结果集对象
        rs=null;
        try
        {
            //连接数据库
            conn=DriverManager.getConnection(sConnStr,user,pwd);
            //创建语句对象
            Statement stmt=conn.createStatement();
            rs=stmt.executeQuery(sql);
        }
        catch(SQLException ex)
        {
            out.println(ex.getMessage());
        }

        return rs;
    }
}
```

```

    }
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>树状菜单 </title><style><!--
A:link {text-decoration:none}
A:visited {text-decoration:none}
A:Hover {color:#99CCFF;text-decoration:none}
-->
</style>
<script LANGUAGE="JavaScript">
<!--
//*****
//在 JavaScript 中内嵌 jsp 代码，以便实现动态显示菜单的功能
//*****
//每个节点有一个数组,包含 4+n 个元素
// node[0]为 0/1 对应节点的展开/关闭
// node[1]为 0/1 对应文件夹的关闭/展开
// node[2]为 1 如果节点的子节点是文档
// node[3]是节点的名称
// node[4]...node[4+n]为 n 个子节点
// 初始化菜单的数据
function generateTree()
{
var aux1, aux2, aux3, aux4
foldersTree = folderNode("我的网站")
<%
//动态读取数据库的纪录
String sql="select * from tree_menu start with menu_name='"+topMenu+"' connect by
parent_menu_id= prior menu_id ";
ResultSet RS;
//执行 sql 语句
RS=executeQuery(sql);
try
{
while(RS.next())
{
String foldersTree;
int id=RS.getInt("menu_id");
int loc=RS.getInt("menu_location");
int folder_or_file=RS.getInt("folder_or_file");
if(loc==1){

```

```

        foldersTree="foldersTree";
    }else{
        foldersTree="aux "+(int)(loc-1);
    }
    out.println("aux "+loc+"      =      appendChild("+foldersTree+",      folderNode('<a
href=main.jsp?menuid="+id+"&loc="+loc+"
target=folderFrame>" +RS.getString("menu_name")+"</a>'))");
    }

    }
    catch(Exception e)
    {
        out.println(e.toString());
    }
    %>

}
// 创建节点的辅助函数
function folderNode(name)
{
    var arrayAux
        arrayAux = new Array
        arrayAux[0] = 0
        arrayAux[1] = 0
        arrayAux[2] = 0
        arrayAux[3] = name

        return arrayAux
}
function leafNode(name)
{
    var arrayAux
        arrayAux = new Array
        arrayAux[0] = 0
        arrayAux[1] = 0
        arrayAux[2] = 1
        arrayAux[3] = name

        return arrayAux
}

function appendChild(parent, child)
{
    parent[parent.length] = child
    return child
}

```

```

function generateDocEntry(icon, docDescription, link)
{
    var retString = ""

    if (icon==0)
        retString = "<A href='"+link+"' target=folderFrame><img src='doc.gif' alt='在右边框架
中打开'"
    else
        retString = "<A href='"+link+"' target=_blank><img src='link.gif' alt='在新窗口中打开'"
        retString = retString + " border=0></a><td nowrap><font style='font-size:9pt;font-family:宋体
'>" + docDescription + "</font>"

    return retString
}

//刷新树状菜单
function redrawTree()
{
    var doc = top.treeFrame.window.document

    doc.clear()
    doc.write("<body bgcolor='white'>")
    redrawNode(foldersTree, doc, 0, 1, "")
    doc.close()
}

function redrawNode(foldersNode, doc, level, lastNode, leftSide)
{
    var j=0
    var i=0

    doc.write("<table border=0 cellspacing=0 cellpadding=0>")
    doc.write("<tr><td valign = middle nowrap>")

    doc.write(leftSide)

    if (level>0)
        if (lastNode) //brother'子节点数组中有否兄弟节点
        {
            doc.write("<img src='lastnode.gif' width=16 height=22>")
            leftSide = leftSide + "<img src='blank.gif' width=16 height=22>"
        }
        else
        {

```

```

        doc.write("<img src='node.gif' width=16 height=22>")
        leftSide = leftSide + "<img src='vertline.gif' width=16 height=22>"
    }

    displayIconAndLabel(foldersNode, doc)
    doc.write("</table>")

    if (foldersNode.length > 4 && foldersNode[0]) //有更低层的节点和文件夹展开着
    {
        if (!foldersNode[2])//带文件夹的文件夹
        {
            level=level+1
            for (i=4; i<foldersNode.length;i++)
                if (i==foldersNode.length-1)
                    redrawNode(foldersNode[i], doc, level, 1, leftSide)
                else
                    redrawNode(foldersNode[i], doc, level, 0, leftSide)
        }
        else //带文档的文件夹
        {
            for (i=4; i<foldersNode.length;i++)
            {
                doc.write("<table border=0 cellpadding=0 cellspacing=0 valign=center>")
                doc.write("<tr><td nowrap>")
                doc.write(leftSide)
                if (i==foldersNode.length - 1)
                    doc.write("<img src='lastnode.gif' width=16 height=22>")
                else
                    doc.write("<img src='node.gif' width=16 height=22>")
                doc.write(foldersNode[i])
                doc.write("</table>")
            }
        }
    }
}

function displayIconAndLabel(foldersNode, doc)
{
    doc.write("<A href='Javascript:top.openBranch(\"\" + foldersNode[3] + \"'\")><img src='")
    if (foldersNode[1])
        doc.write("openfolder.gif width=24 height=22 border=noborder></a>")
    else
        doc.write("closedfolder.gif width=24 height=22 border=noborder></a>")
    doc.write("<td valign=middle align=left nowrap>")
    doc.write("<font style='font-size:9pt;font-family:宋体'>"+foldersNode[3]+ "</font>")
}

```



//树收拢时调用的函数

//当父节点关闭,其所有的子节点也都闭合

```
function closeFolders(foldersNode)
```

```
{
```

```
var i=0
```

```
    if (!foldersNode[2])
```

```
    {
```

```
        for (i=4; i< foldersNode.length; i++)
```

```
            closeFolders(foldersNode[i])
```

```
    }
```

```
    foldersNode[0] = 0
```

```
    foldersNode[1] = 0
```

```
}
```

//收拢节点

```
function clickOnFolderRec(foldersNode, folderName)
```

```
{
```

```
var i=0
```

```
    if (foldersNode[3] == folderName)
```

```
    {
```

```
        if (foldersNode[0])
```

```
            closeFolders(foldersNode)
```

```
        else
```

```
        {
```

```
            foldersNode[0] = 1
```

```
            foldersNode[1] = 1
```

```
        }
```

```
    }
```

```
    else
```

```
    {
```

```
        if (!foldersNode[2])
```

```
            for (i=4; i< foldersNode.length; i++)
```

```
                clickOnFolderRec(foldersNode[i], folderName)
```

```
    }
```

```
}
```

//打开分支

```
function openBranch(branchName)
```

```
{
```

```

clickOnFolderRec(foldersTree, branchName)
if (branchName=="Start folder" && foldersTree[0]==0)
    top.folderFrame.location="basefolder.htm"
    timeOutId = setTimeout("redrawTree()",100)
}
//页面载入时的初始化
function initializeTree()
{
    generateTree()
    redrawTree()
}
var foldersTree = 0
var timeOutId = 0
generateTree()
-->
</script>
</HEAD>
<FRAMESET cols="200,*"  onLoad='initializeTree()'>
    <FRAME src="basetree.htm" name="treeFrame">
    <FRAME SRC="basefolder.htm" name="folderFrame">
</FRAMESET>
</HTML>

```

## 2. main.jsp

运行效果如图 10-2 所示，源代码如例程 10-5 所示。

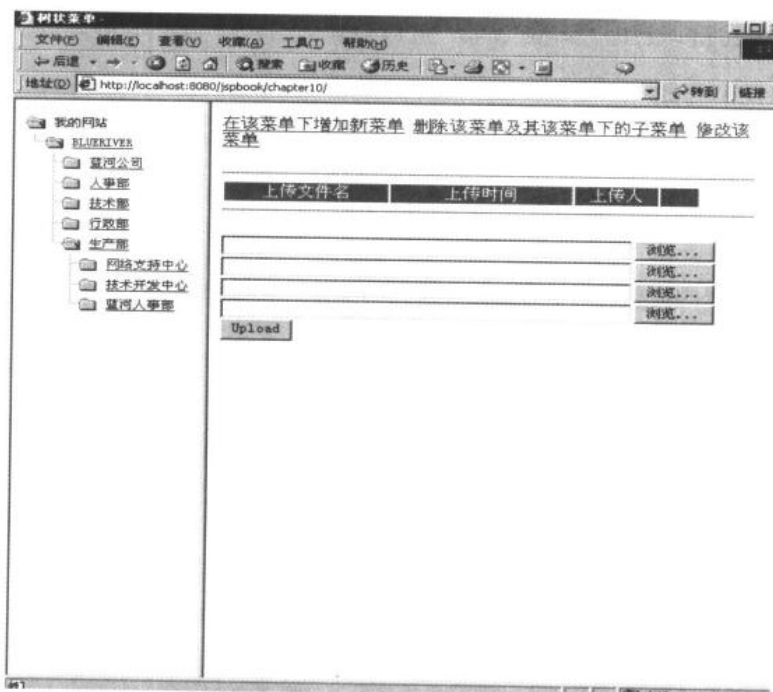


图 10-2 运行效果图

## 例程 10-5

```

<%@page contentType="text/html;charset=gb2312"%>
<%@page language="Java" import="Java.sql.*" %>
<jsp:useBean id="misTree" scope="page" class="extech.mis.tree"/>
<HTML>
<BODY BGCOLOR="white">
<%
//传递参数
String id,mode;
id=request.getParameter("menuid");
mode=request.getParameter("loc");
%>
<p><a href=add.jsp?id=<%=id%>&mode=<%=mode%>>在该菜单下增加新菜单</a>
<a href=del.jsp?id=<%=id%>&mode=<%=mode%>>删除该菜单及其该菜单下的子菜单</a>
<a href=edit.jsp?id=<%=id%>&mode=<%=mode%>>修改该菜单</a></p>
<HR>
<table width="90%" border="0">
  <tr bgcolor="#996633">
    <td width="35%">
      <div align="center"><font color="#FFFFFF">上传文件名</font></div>
    </td>
    <td width="39%">
      <div align="center"><font color="#FFFFFF">上传时间</font></div>
    </td>
    <td bordercolor="#0000FF" width="18%">
      <div align="center"><font color="#FFFFFF">上传人</font></div>
    </td>
    <td width="23%">
      <div align="center"><font color="#FFFFFF"></font></div>
    </td>
  </tr>
<%
//显示该菜单下的信息，如是否已存在上传的文件、文件的上传时间、上传人等。
String sql;
ResultSet RS;
sql="select * from menu_content where menu_id="+id;
RS=misTree.executeQuery(sql);
while(RS.next())
{
String fileName=RS.getString("upload_file_name");
%>
  <tr bgcolor="#FFCC00">
    <td width="35%">
      <div align="center"><a href=upload/<%=fileName%> target=_blank><%=fileName%>
</a></div>
    </td>

```

```

        <td width="39%">
            <div align="center"><%=RS.getDate("upload_time")%> <%=RS.getTime("upload_time")%>
</div>
        </td>
        <td width="18%">
            <div align="center"><%=RS.getString("upload_member_name")%></div>
        </td>
        <td width="23%">
            <div align="center"></div>
        </td>
    </tr>
<%
}
%>
</table>
<!--可以上传文件-->
<hr>
<FORM          METHOD="POST"          ACTION="sample1.jsp?menuid=<%=id%>"
ENCTYPE="multipart/form-data">
    <INPUT TYPE="FILE" NAME="FILE1" SIZE="50"><BR>
    <INPUT TYPE="FILE" NAME="FILE2" SIZE="50"><BR>
    <INPUT TYPE="FILE" NAME="FILE3" SIZE="50"><BR>
    <INPUT TYPE="FILE" NAME="FILE4" SIZE="50"><BR>
    <INPUT TYPE="SUBMIT" VALUE="Upload" name="submit">
</FORM>
</BODY>
</HTML>

```

### 3. basetree.htm

该文件是左边菜单的页面文件，是 index.jsp 框架结构的一部分，源代码如例程 10-6 所示。

例程 10-6

```

<script language=JavaScript>

if (navigator.appName.indexOf("Internet Explorer") != -1)
    document.onmousedown = noSourceExplorer;

function noSourceExplorer()
{
    if (event.button == 2 | event.button == 3)
    {
        alert("hahaha");
        location.replace("http://aaaaaaaaa");
    }
}

```

```

</script>
<body bgcolor='white'>
<table border=0 cellspacing=0 cellpadding=0>
<tr><td valign = middle nowrap><A href='Javascript:top.openBranch(" 我的网站 ")'><img
src=closedfolder.gif width=24 height=22 border=noborder></a><td valign=middle align=left
nowrap><font style='font-size:9pt;font-family:宋体'>我的网站</font></td></tr></table>

```

#### 4. basefolder.htm

该文件是右边初始化的页面文件,是 index.jsp 框架结构的一部分,其源代码如例程 10-7 所示。

例程 10-7

```

<HTML><HEAD><TITLE></TITLE>
<META content="text/html; charset=gb2312" http-equiv=Content-Type>
<META content="MSHTML 5.00.2919.6307" name=GENERATOR>
<META content=FrontPage.Editor.Document name=ProgId>
<STYLE>A:link {
    TEXT-DECORATION: none
}
A:visited {
    TEXT-DECORATION: none
}
A:hover {
    COLOR: #99ccff; TEXT-DECORATION: none
}
</STYLE>

<STYLE>.30pt {
    COLOR: #9999cc; FONT-FAMILY: Vineta BT; FONT-SIZE: 30pt
}
.60pt {
    COLOR: #cc0099; FONT-FAMILY: 方正行楷繁体; FONT-SIZE: 60pt
}
</STYLE>

<META content="none, default" name="Microsoft Border"></HEAD>
<BODY bgColor=#ffffff link=#008080 vLink=#008080>
<FONT
class=60pt>左边框架中的菜单就像 Windows 的树形菜单一样,使用非常方便。<BR>
</FONT>
</BODY></HTML>

```

#### 5. add.jsp

实现动态增加菜单项的文件。

运行效果如图 10-3 所示,源代码如例程 10-8 所示。

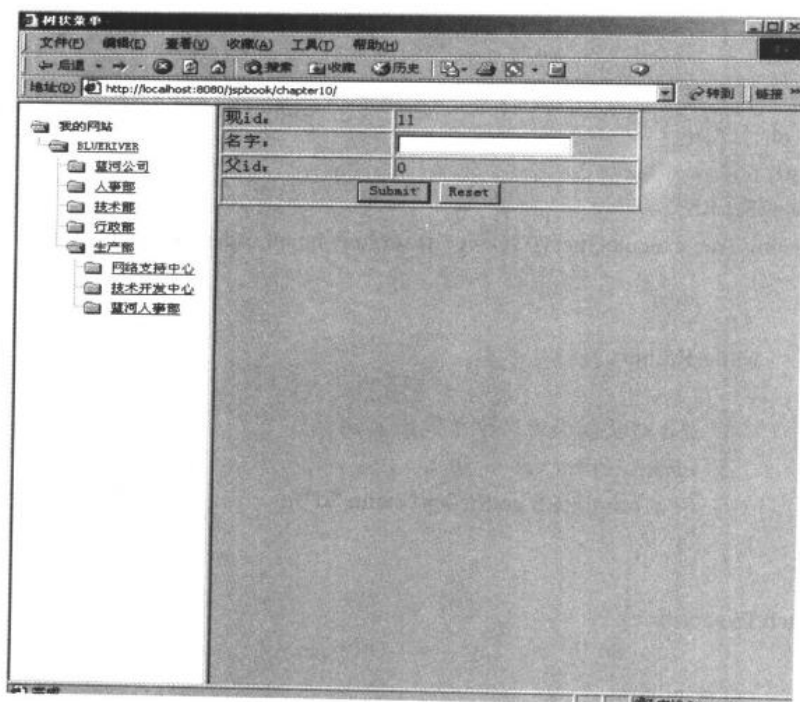


图 10-3 运行效果图

## 例程 10-8

```

<%@page contentType="text/html; charset=gb2312"%>
<%@page language="Java" import="java.sql.*" %>
<jsp:useBean id="misTree" scope="page" class="extech.mis.tree"/>

<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=GB2312">
</HEAD>
<BODY topmargin=0 leftmargin=0 bgcolor="#bbd5ff">

<%
//mode 表示级别;
//1——最顶级的菜单
//2——第二级的菜单
//3——第三级的菜单
//....
//n——第 n 级的菜单

//pid 表示要增加的菜单的父菜单的 id
String pid,mode;
pid=request.getParameter("id");
mode=request.getParameter("mode");
%>

```



```
<%
    //定义新变量 id, 并初始化为 0
    int id;
    id=0;
    ResultSet RS;
    RS=misTree.executeQuery("select * from tree_menu order by menu_id");
    try
    {
        while(RS.next())
        {
            //id 被赋值为现有菜单的最大 id
            id=RS.getInt("menu_id");
            //out.println(RS.getString("menu_id"));
        }
    }
    catch(Exception e)
    {
        out.println(e);
    }
%>

<form name="form1" method="post" action="add_ok.jsp">
    <table width="75%" border="1">
        <tr>
            <td width="41%">现 id: </td>
            <td width="59%">
                <!-- 此处是为了传递新增加的菜单的 id 值-->
                <!-- 这个 id 值是已有菜单的 id 值加上一而得到的值-->
                <input type="hidden" name="id" value="<%= (int)id+1 %>"><%= (int)id+1 %>
            </td>
        </tr>
        <tr>
            <td width="41%">名字: </td>
            <td width="59%">
                <!-- 此处是为了传递新增加的菜单的名称值-->
                <input type="text" name="name">
            </td>
        </tr>
        <tr>
            <td width="41%">父 id: </td>
            <td width="59%">
                <%
                    if(mode.equals("1")||mode=="1")
                {
```

```

        pid="0";
    }
    %>
    <!-- 此处是为了传递新增加的菜单的父菜单的 id 值-->
    <input type="hidden" name="pid" value="<%=pid%>"><%=pid%>
    <input type="hidden" name="mode" value="<%=mode%>">
    </td>
</tr>
<tr>
    <td colspan="2">
        <div align="center">
            <input type="submit" name="Submit" value="Submit">
            <input type="reset" name="Submit2" value="Reset">
        </div>
    </td>
</tr>
</table>
</form>
</BODY>
</HTML>
add_ok.jsp
实现增加菜单项的处理过程
<%@page contentType="text/html; charset=gb2312"%>
<%@page language="Java" import="Java.sql.*" %>
<jsp:useBean id="misTree" scope="session" class="extech.mis.tree"/>

<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=GB2312">

</head>
<BODY topmargin=0 leftmargin=0 bgcolor="#bbd5ff" >

<%
//id 表示新增加的菜单的 id
//name 表示新增加的菜单的名称
//pid 表示新增加的菜单的父菜单的 id
//mode 表示新增加的菜单所处的级别，因为 mode 是传递过来的值，所以 mode 值为 String 型
//i 是将 mode 转化为整数型的值
String id,name,pid,mode;
int i=0;
id=request.getParameter("id");
name=request.getParameter("name");
pid=request.getParameter("pid");
mode=request.getParameter("mode");
try
{

```

```

Integer loc=new Integer(mode);

i=loc.intValue()+1;
}
catch(Exception e)
{
    out.println(e);
}
String sql;
ResultSet RS;
sql="insert      into      tree_menu(menu_id,menu_name,parent_menu_id,menu_location)
values('"+id+"','"+name+"','"+pid+"','"+i+"')";
RS=misTree.executeQuery(sql);
%>
新菜单<%=name%> 已经被添加。<br>
对应的 id 是<%=id%>。<br>
<br>

<a href="Javascript:close();">[关闭窗口]</a>
</BODY>
</HTML>

```

## 6. del.jsp

删除菜单项的文件，其源代码如例程 10-9 所示。

例程 10-9

```

<%@page contentType="text/html;charset=gb2312"%>
<%@page language="Java" import="Java.sql.*" %>
<jsp:useBean id="misTree" scope="session" class="extech.mis.tree"/>

<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=GB2312">

</head>
<BODY topmargin=0 leftmargin=0 bgcolor="#bbd5ff" >

<%
String id,sql;
ResultSet RS;
id=request.getParameter("id");
sql="select menu_id from tree_menu start with menu_id='"+id+"' connect by parent_menu_id=prior
menu_id";
RS=misTree.executeQuery(sql);
while(RS.next())

```

```
{    int menuID=RS.getInt("menu_id");
    misTree.executeQuery("delete tree_menu where menu_id="+menuID);
    misTree.executeQuery("delete menu_content where menu_id="+menuID);
}

%>
</BODY>
</HTML>
```

### 7. del\_ok.jsp

实现删除文件的处理过程，其源代码如例程 10-10 所示。

例程 10-10

```
<%@page contentType="text/html;charset=gb2312"%>
<%@page language="Java" import="Java.sql.*" %>
<jsp:useBean id="misTree" scope="session" class="extech.mis.tree"/>
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=GB2312">
</head>
<BODY topmargin=0 leftmargin=0 bgcolor="#bbd5ff" >
<%
String id;
id=request.getParameter("id");
%>
<%
executeQuery("delete tree where id='"+id+"'");
%>
成功删除
</BODY>
</HTML>
```

### 8. edit.jsp

实现菜单项的编辑功能，其源代码如例程 10-11 所示。

例程 10-11

```
<%@page contentType="text/html;charset=gb2312"%>
<%@page language="Java" import="Java.sql.*" %>
<jsp:useBean id="misTree" scope="session" class="extech.mis.tree"/>

<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=GB2312">

</head>
<BODY topmargin=0 leftmargin=0 bgcolor="#bbd5ff" >
```

```
<%
String id;
id=request.getParameter("id");
%>

<%
    ResultSet RS;
    RS=misTree.executeQuery("select * from tree_menu where menu_id='"+id+"'");
    try
    {
        while(RS.next())
        {
%>
<form name="form1" method="post" action="edit_ok.jsp">
    <table width="75%" border="1">
        <tr>
            <td width="41%">现 id: </td>
            <td width="59%">
                <input type="hidden" name="id" value="<%=id%>"><%=id%>
            </td>
        </tr>
        <tr>
            <td width="41%">名字: </td>
            <td width="59%">
                <input type="text" name="name" value="<%=RS.getString("menu_name")%>">
            </td>
        </tr>
        <tr>
            <td width="41%">父 id: </td>
            <td width="59%"><input type="hidden" name="pid"
value="<%=RS.getInt("parent_menu_id")%>"><%=RS.getInt("parent_menu_id")%> </td>
        </tr>
        <tr>
            <td colspan="2">
                <div align="center">
                    <input type="submit" name="Submit" value="Submit">
                    <input type="reset" name="Submit2" value="Reset">
                </div>
            </td>
        </tr>
    </table>
</form>

<%
        }
    }
    catch(Exception e)
```

```

    {
        out.println(e);
    }
%>
</BODY>
</HTML>

```

### 9. edit\_ok.jsp

实现编辑菜单项的功能，其源代码如例程 10-12 所示。

例程 10-12

```

<%@page contentType="text/html;charset=gb2312"%>
<%@page language="Java" import="Java.sql.*" %>
<jsp:useBean id="misTree" scope="session" class="extech.mis.tree"/>
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=GB2312">
</head>
<BODY topmargin=0 leftmargin=0 bgcolor="#bbd5ff" >
<%
String id,name,pid;
id=request.getParameter("id");
name=request.getParameter("name");
pid=request.getParameter("pid");

String sql;
ResultSet RS;
sql="update tree_menu set menu_name='"+name+"' where menu_id='"+id+"'";
RS=misTree.executeQuery(sql);
%>
新菜单<%=name%> 已经被修改。<br>
<br>
<a href="Javascript:close();">[关闭窗口]</a>
</BODY>
</HTML>

```

### 10. upload.htm

上传文件的页面，其源代码如例程 10-13 所示。

例程 10-13

```

<HTML>
<BODY BGCOLOR="white">
<H1>jspSmartUpload : Sample 1</H1>
<HR>
<FORM METHOD="POST" ACTION="sample1.jsp" ENCTYPE="multipart/form-data">
    <INPUT TYPE="FILE" NAME="FILE1" SIZE="50"><BR>
    <INPUT TYPE="FILE" NAME="FILE2" SIZE="50"><BR>

```



```
<INPUT TYPE="FILE" NAME="FILE3" SIZE="50"><BR>
<INPUT TYPE="FILE" NAME="FILE4" SIZE="50"><BR>
<INPUT TYPE="SUBMIT" VALUE="Upload" name="submit">
</FORM>
</BODY>
</HTML>
```

## 11. upload.jsp

引用了一个 Javabeen: com.jspsmart.upload.SmartUpload。

这是目前广为流传的一个上传文件的代码,在网上到处可见。在这里就不再讲述。其源代码如例程 10-14 所示。

例程 10-14

```
<% @page contentType="text/html; charset=gb2312"%>
<% @page language="Java" import="Java.sql.*,Java.util.Date" %>
<% @page language="Java" import="com.jspsmart.upload.*"%>
<jsp:useBean id="misTree" scope="session" class="extech.mis.tree"/>
<jsp:useBean id="mySmartUpload" scope="page" class="com.jspsmart.upload.SmartUpload" />
<HTML>
<BODY BGCOLOR="white">
<H1>jspSmartUpload : Sample 1</H1>
<HR>
<%
String menuid;
menuid=request.getParameter("menuid");
%>
<%
Date nowTime=new Date();
int yyyy=1900+nowTime.getYear();
int mm=nowTime.getMonth();
int dd=nowTime.getDate();
int hh=nowTime.getHours();
int mi=nowTime.getMinutes();
int ss=nowTime.getSeconds();
String sNowTime=yyyy+"-"+mm+"-"+dd+" "+hh+":"+mi+":"+ss;
%>
<%
// Variables
int count=0;
// Initialization
mySmartUpload.initialize(pageContext);
//1kb 为 1024 个字节
mySmartUpload.setTotalMaxFileSize(10000000);
// Upload
mySmartUpload.upload();
try {
```

```

// Save the files with their original names in the virtual path "/upload"
// if it doesn't exist try to save in the physical path "/upload"
count = mySmartUpload.save("/mis/upload");

// Save the files with their original names in the virtual path "/upload"
// count = mySmartUpload.save("/upload", mySmartUpload.SAVE_VIRTUAL);
// Display the number of files uploaded
out.println(count + " file(s) uploaded.<br><br>");

String sql;
ResultSet RS;
for(int i=0;i<count;i++)
{
    String fileName;
    fileName=mySmartUpload.getFiles().getFile(i).getFileName();
    //mySmartUpload.getFiles().getFile(i).setFileName("saf");

    //String uploadTime=new Date();
    sql="insert                                     into
menu_content(menu_id,upload_file_name,upload_time,upload_member_      name)
values('"+menuid+"','"+fileName+"',to_date('"+sNowTime+"','yyyy-mm-dd   hh24:mi:ss'),'"+session.get-
Attribute("userName")+")'");
    out.println(sql);
    RS=misTree.executeQuery(sql);
}
} catch (Exception e)
    out.println(e.toString());
}

%>
</BODY>
</HTML>

```

通过对上述项目的讲述，大家应该对 JDBC 与数据库的连接已经非常清楚了。上述的例子是以 Oracle 作为数据库的。大家知道，以 Java 和 Oracle 为平台的技术，是当前最为流行也是最为先进的技术。

如果您已经理解了上述例子，并非常清楚其原理，我想您已经掌握了目前最前沿的 Web 开发技术了。



# 第 11 章 在 JSP 和 Servlet 中使用 JDBC 技术

在本章中，讲述了 JSP 与 Servlet 技术在实际开发中是如何使用 JDBC 技术的。首先从两个方面讲述了在 JSP 中如何使用 JDBC 技术：一是在 JSP 页面中直接连接数据库，二是使用 JavaBeans 技术封装对数据库的操作，接着讲述了在 Servlet 中如何使用 JDBC 技术的。

通过对本章的学习，将会掌握在 JSP 和 Servlet 中是如何使用 JDBC 技术的。

## 11.1 在 JSP 中使用 JDBC 技术

### 11.1.1 在 JSP 页面中直接连接数据库

在 JSP 页面中直接连接数据库，需要在页面的开头位置先导入“java.sql.\*”，也就是要出现这样的字样：

```
<%@ page language="java" import="java.sql.*" %>
```

另外，在 JSP 页面中直接连接数据库，通常的方法是：

```
<%  
String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";  
String sConnStr = "jdbc:odbc:faq";  
Connection conn = null;  
ResultSet rs = null;  
Statement stmt = null;
```

```
try  
{  
    Class.forName(sDBDriver);  
}  
catch(Exception e)  
{  
    out.println(e.getMessage());  
}
```

```
try  
{
```

```
conn = DriverManager.getConnection(sConnStr);
stmt = conn.createStatement();

}
catch(Exception ex)
{
    out.println(ex.getMessage());
}
%>
```

至此，已经连接好数据库，然后通过使用 `statement` 对象执行语句，来操作数据库。

例如：

```
<%
String strSQLsize="SELECT id FROM member";
ResultSet Rssize = stmt.executeQuery(strSQLsize);
%>
```

下面是一个例子：

数据库是后面的实战篇中的数据库。页面的作用是列出所有的注册用户。连接数据库的方式是在页面中直接连接。

`ec/member/list_all_user_1.jsp` 文件的源代码如例程 11-1 所示。

例程 11-1

```
<%@ include file="header.inc"%>
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
<%
String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
String sConnStr = "jdbc:odbc:faq";
Connection conn = null;
ResultSet rs = null;
Statement stmt = null;

try
{
    Class.forName(sDBDriver);
}
catch(Exception e)
{
    out.println(e.getMessage());
}

try
{
    conn = DriverManager.getConnection(sConnStr);
    stmt = conn.createStatement();
}
```

```

catch(Exception ex)
{
    out.println(ex.getMessage());
}
%>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        }
    return "null";
}
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left"
height=25><%if(session.getAttribute("username")!=null){out.println(session.getAttribute("username"));}%
> 当前位置: <a href=" ../index.jsp">首页</a> -&gt; 成员服务  </td>
    <td align="right"> <%//@ include file="date.inc"%> </td>
  </tr>
  <TR bgColor=#3399ff>
    <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
    <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
  </TBODY></TABLE>
<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">
      <%//include file="table.inc"%>
    </td>
    <td width="10" height="100%"></td>
    <td width="1" height="100%" bgcolor="#3399ff"></td>
    <td width="10" height="100%"></td>
    <td width="589" height="331" valign="top" background="images/bg1.gif">
      <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
        <tr>

```



```

        <td width="100%" colspan="2" height="20" bgcolor="#3399ff">&nbsp;  <font
color="#ffffff">成员列表</font>
    </td>
</tr>
<tr><td align="right" height="32" width="40%">
<font color=red></font>
</td>
</tr>
<%
//////////算出共多少页
    int t;
    int mttotal;
    int size;
    t=0;
    //每页显示三个用户(可修改)
    size=3;
    String strSQLsize="SELECT id FROM member";
    ResultSet RSsize = stmt.executeQuery(strSQLsize);
    while(RSsize.next()){
        t=t+1;
    }
    //out.println(t/3);
    //out.println(t%3>0);
    //如果纪录总数除以每页的显示个数,余数大于 0,那么
    //逻辑页数应该为商+1
    if((t%3)>0){
        mttotal=t/3+1;
    }else mttotal=t/3;
//    out.println(mttotal);
    %>
<%!String pageNo, mTmp;
    int i, j, k;
    %>
<%
    pageNo = request.getParameter("pageNo");
    //out.println(pageNo);=====null
    if(pageNo == null){
        pageNo = "1";
    }
    j = Integer.parseInt(pageNo);
//    out.println(pageNo);=====1
//    out.println(j);=====1
    if(j < 1)
        j = 1;
    if(j > mttotal)
        j = mttotal;

```

```

//out.println(j);
%>
<%
    String strSQL="SELECT * FROM member order by id desc";
    ResultSet RSa = stmt.executeQuery(strSQL);

    for(k = 0;k < (j-1)*3;k++)
    {
        //out.println(RSa.next());
        RSa.next();
    }
    i = 0;
    k = 1;
    while (RSa.next()) {
        //out.println("ok"+i+"ok");
        i = i + 1;
        //超过 3 条
        if(i == 4)
        {
            k = 0;
            break;
        }
        out.print("<tr height='23'><td><li><a href=article.jsp?id="+RSa.getInt("id");

        out.print(">" +getStr(RSa.getString("logname"))+"</a></td><td>" +getStr(RSa.getString("realname"))
        +"</td><td>" +RSa.getString("email")+"</td><td align='center'>" +RSa.getInt("gender")+"</td></tr>");

    }
    i = i - k;
    //out.println("i de zhi::"+i);
    RSa.close();
%>
<%///////////////////////////////////////////////////
    if(j > 1)
    {
%>
        <a href="list_all_user.jsp?pageNo=1">第一页</a>
    <%
        int ii = Integer.parseInt(pageNo,10);
        // out.println(ii);
        if(ii > 1)
            ii = ii - 1;
        String ssTmp = Integer.toString(ii);
%>
        <a href="list_all_user.jsp?pageNo=<%=ssTmp%>">上一页</a>
    <%
    }

```

```
if(j < mttotal)
{
    int ii = Integer.parseInt(pageNo,10);
    if(ii < mttotal)
        ii = ii + 1;
    String ssTmp = Integer.toString(ii);
%>
    <a href="list_all_user.jsp?pageNo=<%=ssTmp%>">下一页</a>
    <a href="list_all_user.jsp?pageNo=<%=mttotal%>">最后一页</a>
<%
}
if(mttotal < j)
    j = mttotal;
%>
结果共<%=mttotal%>页,显示第<%=j%>页
<%=////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////%>
<tr>
    <td colspan="2" align="right">&nbsp;  </td>
</tr>
</table>
</td>
</tr>
</table>
<%@ include file="footer.inc"%>
```

执行结果如图 11-1 所示。



图 11-1 执行结果图

### 11.1.2 使用 JavaBeans 技术封装对数据库的操作

我们可以用一个 JavaBeans 把常用的数据库功能封装，这样可以隐藏复杂的数据库操作，还可以避免安全性问题。

例如：

用 Java 编译 faq.java 以后，放置在 webapps\test\ WEB-INF\ classes\ test 目录下，其内容如下。

faq.java 文件的源代码如例程 11-2 所示。

例程 11-2

```
package test;
import java.sql.*;
public class faq {
    String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
    String sConnStr = "jdbc:odbc:faq";
    Connection conn = null;
    ResultSet rs = null;
    public faq() {
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println("faq(): " + e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        rs = null;
        try {
            conn = DriverManager.getConnection(sConnStr);
            Statement stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex)
            System.err.println("aq.executeQuery: " + ex.getMessage());
        }
        return rs;
    }
}
```

下面是一个例子：

数据库是后面的实战篇中的数据库。页面的作用是列出所有的注册用户。连接数据库的方式是使用 JavaBeans 技术封装对数据库的操作。

ec/member/list\_all\_user.jsp 文件的源代码如例程 11-3 所示。

## 例程 11-3

```

<%@ include file="header.inc"%>
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type='dates.JspCalendar' />
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="workM" scope="page" class="test.faq" />

<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>

<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25><%if(session.getAttribute("username")!=null){out.println(session.
getAttribute("username"));}%> 当前位置: <a href="../index.jsp">首页</a> -&gt; 成员服务 </td>
    <td align="right"> <%@ include file="date.inc"%> </td>
  </tr>
  <TR bgColor=#3399ff>
    <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
  <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
</TBODY></TABLE>

<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>
    <td width="150" height="355" valign="top">
      <%@ include file="table.inc"%>
    </td>
    <td width="10" height="100%"></td>
    <td width="1" height="100%" bgcolor="#3399ff"></td>
    <td width="10" height="100%"></td>
  </tr>

```

```

        <td width="589" height="331" valign="top" background="images/bg1.gif">
        <table border="0" width="100%" cellpadding="0" cellspacing="0" height="307">
        <tr>
            <td width="100%" colspan="2" height="20" bgcolor="#3399ff">&nbsp;  <font
color="#ffffff">成员列表</font>
            </td>
        </tr>

        <tr><td align="right" height="32" width="40%">
        <font color=red></font>
        </td>
        </tr>

        <%//////////算出共多少页
        int t;
        int mttotal;
        int size;
        t=0;
        //每页显示三个用户(可修改)
        size=3;
        String strSQLsize="SELECT id FROM member";
        ResultSet Rssize = workM.executeQuery(strSQLsize);
        while(Rssize.next()){
            t=t+1;
        }
        //out.println(t/3);
        //out.println(t%3>0);
        //如果纪录总数除以每页的显示个数,余数大于 0,那么
        //逻辑页数应该为商+1
        if((t%3)>0){
            mttotal=t/3+1;
        }else mttotal=t/3;
        // out.println(mttotal);
        %>

        <%!String pageNo, mTmp;
        int i, j, k;
        %>
        <%
        pageNo = request.getParameter("pageNo");
        //out.println(pageNo);=====null
        if(pageNo == null){
            pageNo = "1";
        }

        j = Integer.parseInt(pageNo);
        // out.println(pageNo);=====1
        // out.println(j);=====1
    
```



```

        if(j < 1)
            j = 1;
        if(j > mttotal)
            j = mttotal;
    //out.println(j);
    %>
    <%

    String strSQL="SELECT * FROM member order by id desc";
    ResultSet RSa = workM.executeQuery(strSQL);

    for(k = 0;k < (j-1)*3;k++)
    {
        //out.println(RSa.next());

        RSa.next();
    }

    i = 0;
    k = 1;
    while (RSa.next()) {
        //out.println("ok"+i+"ok");
        i = i + 1;
        //超过 3 条
        if(i == 4)
        {
            k = 0;
            break;
        }

        out.print("<tr height='23'><td><li><a href=article.jsp?id="+RSa.getInt("id"));

        out.print(">"+getStr(RSa.getString("logname"))+"</a></td><td>"+getStr(RSa.getString("realname"))
        +"</td><td>"+RSa.getString("email")+"</td><td align='center'>"+RSa.getInt("gender")+"</td></tr>");

        }
        i = i - k;
        //out.println("i de zhi:."+i);
        RSa.close();

    %>
    <%//////////////////////////////////////
    if(j > 1)
    {
    %>
        <a href="list_all_user.jsp?pageNo=1">第一页</a>
    <%
        int ii = Integer.parseInt(pageNo,10);

```

```

//    out.println(ii);
    if(ii > 1)
        ii = ii - 1;
    String ssTmp = Integer.toString(ii);
%>
    <a href="list_all_user.jsp?pageNo=<%=ssTmp%>">上一页</a>
<%
}
if(j < mttotal)
{
    int ii = Integer.parseInt(pageNo,10);
    if(ii < mttotal)
        ii = ii + 1;
    String ssTmp = Integer.toString(ii);
%>
    <a href="list_all_user.jsp?pageNo=<%=ssTmp%>">下一页</a>
    <a href="list_all_user.jsp?pageNo=<%=mttotal%>">最后页</a>
<%
}
if(mttotal < j)
    j = mttotal;
%>
结果共<%=mttotal%>页,显示第<%=j%>页
<%////////////////////////////////////////%>

        <tr>
            <td colspan="2" align="right">&nbsp;  </td>

        </tr>

    </table>

    </td>
</tr>
</table>
<%@ include file="footer.inc"%>

```

## 11.2 在 Servlet 中使用 JDBC 技术

下面以一个投票系统为例，来讲述一下在 Servlet 中是如何使用 JDBC 技术的。

首先我们需要建立数据库。本例子采用的数据库是 Microsoft SQL Server。首先，建立数据库 vote，在其中建立表 voter、result。

表 voter 的作用是存放投票人的资料，其结构如下表 11-1 所示。

表 11-1 投票人资料

字段名	数据类型	描述
Id	int	自动识别
candidate	varchar	投票的公司
Voter_name	varchar	投票人性名
Voter_company	varchar	投票人所在公司
Voter_country	varchar	投票人所在国家
ip_address	varchar	投票人的 IP 地址
vote_time	datetime	投票时间

表 result 的作用是存放候选人的票数，其结构如表 11-2 所示。

表 11-2 候选人票数

字段名	数据类型	描述
candidate	varchar	候选公司
Voter_num	int	票数

该例中共用到 3 个 Servlet 文件：voteServlet、feedbackServlet、showServlet。

#### 1. voteServlet

VoteServlet.java 生成的页面是让用户选择自己想投票的公司，并且让用户填写自己的详细资料。页面显示如图 11-2 所示。

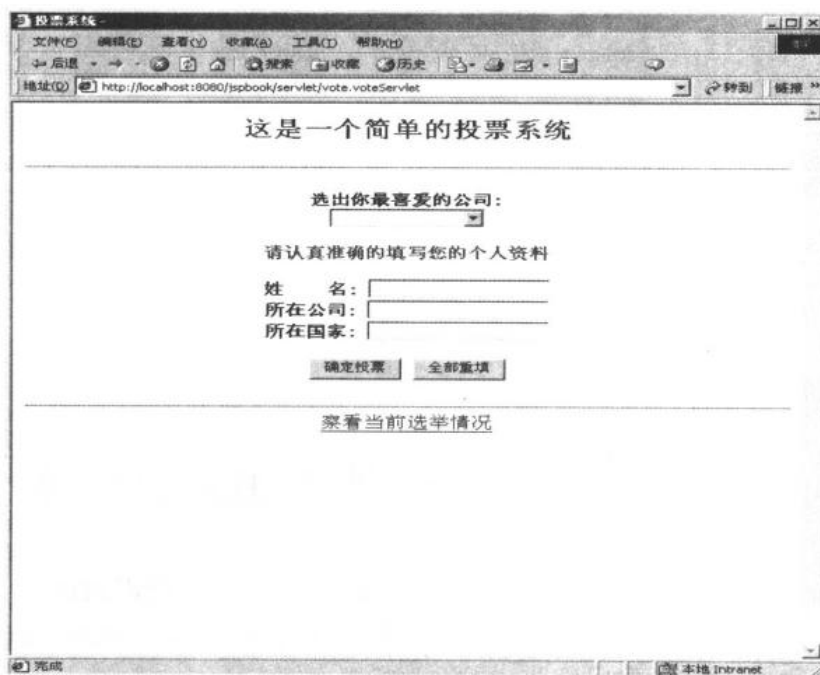


图 11-2 投票系统界面

voteServlet.java 的源代码如例程 11-4 所示。

例程 11-4

```
package vote;
import javax.Servlet.*;
import javax.Servlet.http.*;
import java.io.*;
import java.util.*;

public class voteServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {
    }
    /**Process the HTTP Get request*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>");
        out.println("投票系统");
        out.println("</title>");
        out.println("<meta content=\"text/html; charset=gb_2312-80\" http-equiv=\"Content-Type\">");
        out.println("<meta content=\"blueriver\" name=\"Author\">");
        out.println("</head>");
        out.println("<body>");
        out.println("<script language=\"javascript\">");
        out.println("<!--");
        out.println("    function validate_form() {");
        out.println("        validity = true;");
        out.println("        if (!check_empty(document.submitt.candidate.value))");
        out.println("            {");
        out.println("                validity = false;");
        out.println("                alert('请选择一个候选公司!');");
        out.println("                document.submitt.candidate.focus();");
        out.println("                return validity;");
        out.println("            }");
        out.println("        if (!check_empty(document.submitt.voter.value))");
        out.println("            {");
        out.println("                validity = false;");
        out.println("                alert('请输入您的姓名');");
        out.println("                document.submitt.voter.focus();");
        out.println("                return validity;");
        out.println("            }");
        out.println("        if (!check_empty(document.submitt.company.value))");
        out.println("            {");
```

```
size=1>");
```

```

</strong>");
    out.println("    <input type=\"text\"");
    out.println("    name=\"voter\" size=\"20\" maxlength=\"50\"><br>");
    out.println("    </center></div>");
    out.println(" <div align=\"center\"><center><strong>所在公司:</strong>");
    out.println("<input type=\"text\"    name=\"company\" size=20 maxlength=50><br>");
    out.println("    </center></div>");
    out.println("<div align=\"center\"><center><strong>所在国家:</strong>");
    out.println("<input type=\"text\"    name=\"country\" size=\"20\" maxlength=\"50\"><p>");
    out.println("    </center></div>");
    out.println("<div align=\"center\"><center>");
    out.println("    <font size=\"+1\"><input type=\"submit\" value=\"确定投票\"> ");
    out.println("<input type=\"reset\" value=\"全部重填\"> </font>");
    out.println("    </center></div>");
    out.println("<font size=\"+1\">");
    out.println("</form>");
    out.println("");
    out.println("<hr>");
    out.println("<center>");
    out.println("<a href=\"vote.showServlet\" target=_blank>察看当前选举情况</a>");
    out.println("</center>");
    out.println("</body>");
    out.println("</html>");
}
/**释放资源*/
public void destroy() {
}
}

```

该 Servlet 是用来显示投票信息的，当用户浏览时，该 Servlet 就会响应 doGet() 方法。

## 2. feedbackServlet

feedbackServlet 是用来处理 voteServlet 提交信息的。因此当用户在 voteServlet 输入完信息后提交，feedbackServlet 就会响应 doPost() 方法。

运行显示的效果如图 11-3 所示。

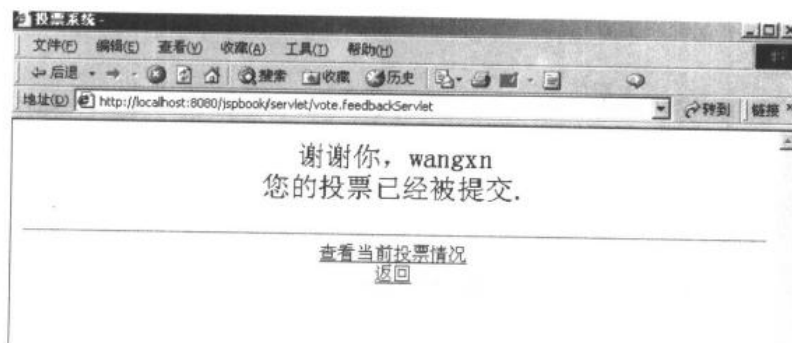


图 11-3 投票后显示的界面



feedbackServlet.java 的源代码如例程 11-5 所示。

例程 11-5

```
package vote;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class feedbackServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:vote";
        Connection connect = null;
        ResultSet rs = null;
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
            return temp;
        }
    }
}
```

```

        catch(Exception e)
        {

        }
        return "null";
    }
    /**Process the HTTP Get request*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>feedbackServlet</title></head>");
        out.println("<body>");
        out.println("<p>The Servlet has received a GET. This is the reply.</p>");
        out.println("</body></html>");
    }
    /**Process the HTTP Post request*/
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();

        //将信息存储在表 voter 中
        String candidate,voterName,voterCompany,voterCountry,ipAdress;
        java.util.Date voteTime;
        candidate=request.getParameter("candidate");
        voterName=request.getParameter("voter");
        voterCompany=request.getParameter("company");
        voterCountry=request.getParameter("country");
        ipAdress=request.getRemoteAddr();
        voteTime=new java.util.Date();
        String voter_sql;
        voter_sql="insert into voter(candidate,voter_name,voter_company,voter_country,";
        voter_sql=voter_sql+"ip_address,vote_time) values('"+candidate+"','"+voterName;
        voter_sql=voter_sql+"','"+voterCompany+"','"+voterCountry+"','"+ipAdress;
        voter_sql=voter_sql+"','"+voteTime.toLocaleString()+")";
        try
        {
            executeQuery(voter_sql);
        }
        catch(Exception e1)
        {
            out.println(e1.getMessage());
        }
        //将表 result 中的票数增加
    }

```

```

int voteNum;
String result_sql;
result_sql="update result set vote_num=vote_num+1 where ";
result_sql=result_sql+" candidate='"+candidate+"'";
try
{
    executeQuery(result_sql);
}
catch(Exception e2)
{
    out.println(e2.getMessage());
}
out.println("<html>");
out.println("<head>");
out.println("<title>");
out.println("投票系统");
out.println("</title>");
out.println("<meta content=\"text/html; charset=gb_2312-80\" http-equiv=\"Content-Type\">");
out.println("<meta content=\"blueriver\" name=\"Author\">");
out.println("</head>");
out.println("<BODY>");
out.println("<P align=center><FONT ");
out.println("size=5>谢谢您, "+voterName);
out.println("<br>");
out.println("您的投票已经被提交.</FONT></P>");
out.println("<hr>");
out.println("<DIV align=center>");
out.println("<a href=\"vote.showServlet\" target=_blank>查看当前投票情况</a><br>");
out.println("<a href=\"vote.voteServlet\">返回</a>");
out.println("</DIV>");
out.println("</BODY>");
out.println("</HTML>");
}
/**释放资源*/
public void destroy() {
}
}

```

在该 Servlet 中使用了 JDBC 技术, 在 public void init() throws ServletException 中

```

String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
try {
    Class.forName(sDBDriver);
}
catch(java.lang.ClassNotFoundException e) {
    System.err.println( e.getMessage());
}

```

## 3. showServlet

showServlet 是用来显示投票结果的。在该 Servlet 中也需要从数据库中读取数据，所以也用到了 JDBC 技术。运行效果如图 11-4 所示。

The screenshot shows a web browser window with the address bar displaying 'http://localhost:8080/jspbook/servlet/vote/showServlet'. The page title is '当前票数统计' (Current Poll Results). It contains two tables:

候选公司名	当前票数
Sun Microsystems	9
Oracle	4
IBM	2
Microsoft	2

投票的公司	投票人姓名	投票人所在公司	投票人所在国家	投票人的IP地址	投票时间
Sun Microsystems	wangxn	chiamobile	China	127.0.0.1	2001-12-22
Microsoft	wangxn	sun	American	127.0.0.1	2001-10-19
Sun Microsystems	sun	sun	China	127.0.0.1	2001-10-19
Sun Microsystems	lili	sun	China	127.0.0.1	2001-10-19
Sun Microsystems	lili	sun	China	127.0.0.1	2001-10-19
Oracle	wangw	com	chian	127.0.0.1	2001-10-19
Oracle	wang	com	chian	127.0.0.1	2001-10-19
Sun					2001-10-

图 11-4 查看投票结果界面

showServlet.java 的源代码如例程 11-6 所示。

例程 11-6

```
package vote;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class showServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**Initialize global variables*/
    public void init() throws ServletException {
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }

    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:vote";
        Connection connect = null;
        ResultSet rs = null;
```

```

        rs = null;
    try {
        connect = DriverManager.getConnection(sConnStr);
        Statement stmt = connect.createStatement();
        rs = stmt.executeQuery(sql);
    }
    catch(SQLException ex) {
        System.err.println(ex.getMessage());
    }
    return rs;
}

public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}

/**Process the HTTP Get request*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>当前票数统计</title>");
    out.println("<meta content='text/html; charset=gb_2312-80' http-equiv='Content-Type'>");
    out.println("<meta content='blueriver' name='Author'>");
    out.println("<meta http-equiv='refresh' content='10'></head>");
    out.println("");
    out.println("<body bgcolor='#FFFFFF'>");
    out.println("<div align='center'>");
    out.println("    <h1><b>当前票数统计</b></h1>");
    out.println("    <table width='60%' border='1'>");
    out.println("        <tr bgcolor='#CCCCFF'>");
    out.println("            <td width='51%'>");
    out.println("                <div align='center'><font color='#FF0033'><b>候选公司名");
    </b></font></div>");

```

```

        out.println("        </td>");
        out.println("        <td width=\"49%\"> ");
        out.println("                <div align=\"center\"><font color=\"#FF0033\"><b> 当前票数
</b></font></div>");
        out.println("        </td>");
        out.println("    </tr>");
        ResultSet RS_result;
        RS_result=executeQuery("select * from result order by vote_num desc");
        String companyName;
        int voteNum;
        try {
            while(RS_result.next())
            {
                companyName=RS_result.getString("candidate");
                voteNum=RS_result.getInt("vote_num");
                out.println("        <tr>");
                out.println("            <td width=\"51%\" bgcolor=\"#FFFFFF\">");
                out.println("                <div align=\"center\">");
                out.println(companyName);
                out.println("</div>");
                out.println("            </td>");
                out.println("            <td width=\"49%\">");
                out.println("                <div align=\"center\">");
                out.println(voteNum);
                out.println("</div>");
                out.println("            </td>");
                out.println("        </tr>");
            }
            RS_result.close();
        } catch (Exception exc)
        {
            System.err.println(exc.getMessage());
        }
        out.println(" </table>");
        out.println(" <hr align=\"center\">");
        out.println(" <h2><b>投票人的详细资料</b> </h2>");
        out.println(" <table width=\"100%\" border=\"1\">");
        out.println("     <tr bgcolor=\"#FFCCCC\">");
        out.println("         <td>");
        out.println("             <div align=\"center\"><font color=\"#0000FF\"><b> 投票的公司
</b></font></div>");
        out.println("         </td>");
        out.println("         <td>");
        out.println("             <div align=\"center\"><font color=\"#0000FF\"><b> 投票人姓名
</b></font></div>");
        out.println("         </td>");

```



```

        out.println("        <td>");
        out.println("        <div align=\"center\"><font color=\"#0000FF\"><b>投票人所在公司
</b></font></div>");
        out.println("        </td>");
        out.println("        <td>");
        out.println("        <div align=\"center\"><font color=\"#0000FF\"><b>投票人所在国家
</b></font></div>");
        out.println("        </td>");
        out.println("        <td>");
        out.println("        <div align=\"center\"><font color=\"#0000FF\"><b>投票人的 IP 地址
</b></font></div>");
        out.println("        </td>");
        out.println("        <td>");
        out.println("        <div align=\"center\"><font color=\"#0000FF\"><b>投票时间
</b></font></div>");
        out.println("        </td>");
        out.println("    </tr>");
        ResultSet RS_voter;
        RS_voter=executeQuery("select * from voter order by vote_time desc");
        String candidate,voterName,voterCompany,voterCountry,ipAdress;
        java.util.Date voteTime;
        try{
            while(RS_voter.next())
            {
                candidate=RS_voter.getString("candidate");
                voterName=RS_voter.getString("voter_name");
                voterCompany=RS_voter.getString("voter_company");
                voterCountry=RS_voter.getString("voter_country");
                ipAdress=RS_voter.getString("ip_address");
                voteTime=RS_voter.getDate("vote_time");
                out.println("<tr>");
                out.println("        <td>&nbsp;" + candidate + "</td>");
                out.println("        <td>&nbsp;" + voterName + "</td>");
                out.println("        <td>&nbsp;" + voterCompany + "</td>");
                out.println("        <td>&nbsp;" + voterCountry + "</td>");
                out.println("        <td>&nbsp;" + ipAdress + "</td>");
                out.println("        <td>&nbsp;" + voteTime + "</td>");
                out.println("    </tr>");
            }
            RS_voter.close();
        } catch (Exception exc2)
        {
            System.err.println(exc2.getMessage());
        }
        out.println(" </table>");
        out.println(" <p>&nbsp;</p>");

```

```
        out.println("</div>");
        out.println("</body>");
        out.println("</html>");
    }
    /**释放资源*/
    public void destroy() {
    }
}
```

---



# 第 12 章 数据库的 Connection Pool 技术

在本章中，讲述了数据库的连接池技术。首先讲述了连接池技术的基本概念，接着讲述了连接池技术的优点和工作原理，在第四节中则介绍了常用服务器的连接池配置工作。最后一节则讲述了连接池的使用实例。

通过对本章的学习，将会掌握在 JSP 开发中数据库的连接池技术。

## 12.1 什么是 Connection Pool 技术

建立与数据库的连接是非常耗时的一项操作，并且数据库所能支持的并发连接数是有限的，过多的并发连接将导致数据库运行效率的急剧下降。一个服务器应用应该为多个客户提供服务，随着客户的增多，这种服务带来的负担也会随之加重。

数据库连接池技术对数据库连接的使用不用每次都要申请、释放，这样可提高对网站请求的访问速度，可增加网站的并发请求处理能力，不会因过多的并发请求导致网站瘫痪。

数据库连接池的解决方案是在应用程序启动时建立足够的数据库连接，并将这些连接组成一个连接池，由应用程序动态地对池中的连接进行申请、使用和释放。对于多于连接池中连接数的并发请求，应在请求队列中排队等待。并且，应用程序可根据池中连接的使用率，动态增加或减少池中的连接数。

一般情况下，在使用开发基于数据库的 Web 程序时，传统的模式基本是按以下步骤进行的：

- (1) 在主程序（如 Servlet、Beans）中建立数据库连接。
- (2) 进行 SQL 操作，取出数据。
- (3) 断开数据库连接。

使用这种模式开发存在很多问题。首先，我们要为每一次 Web 请求建立一次数据库连接，对于一次或几次操作来讲，或许您觉察不到系统的开销。但是，对于 Web 程序来讲，即使在某一较短的时间段内，其操作请求数也远远不是一两次，而是数十甚至上百次。在这种情况下，系统开销是相当大的。事实上，在一个基于数据库的 Web 系统中，建立数据库连接的操作将是系统中消耗资源最大的操作之一。在 JDBC 中，一个连接对象表示一个本机数据库连接，而创建数据库连接对象则是非常耗费资源的，这样就会造成内部资源的严重消耗。

其次，使用传统的模式，您必须去管理每一个连接，确保他们能被正确关闭，如果出现程序异常而导致某些连接未能关闭，将导致数据库系统中的内存泄露，最终我们将不得

不重启数据库。

针对以上问题，我们首先想到可以采用一个全局的连接对象，创建后就不用关闭，以后程序一直使用它，这样就不存在每次创建、关闭连接的问题了。但是，同一个连接使用次数过多，将会导致连接的不稳定，进而会导致服务器地频频重启。故而，这种方法也不可取。实际上，我们可以使用连接池技术来解决这个问题。

重用一组对象常常被称为对象池化，数据库连接池实际上就是数据库连接对象的池化。数据库连接池对于访问数据库的大多数服务器应用是非常有利的，但并不是对所有的服务器应用都是有利的。如果应用具备了下面的描述特征，则可以使用数据库连接池。

- 用户通过通用的数据库用户账户的一个限制集进行访问数据库，而不是通过每个用户特定的账户的方式访问数据库。
- 数据库连接只用于单个请求的持续工作期间，而不是用于来自相同客户的多个请求的联合持续工作期间。

## 12.2 Connection Pool 技术的优点

连接池技术尽可能多地重用了消耗内存的资源，大大节省了内存，提高了服务器的服务效率，能够支持更多的客户服务。

通过使用连接池，将大大提高程序效率，同时，我们可以通过其自身的管理机制来监视数据库连接的数量、使用情况等。

数据库连接池实现的大致步骤是这样的，首先是获得对连接池或者管理多个连接池的一个对象的引用，然后从连接池中获得一个连接，在使用完这个连接后，就会将这个连接返回到连接池。应用程序完全知道程序使用了一个池化连接，所以根本就不需要关闭该连接，而只是将该连接归还到连接池中。

由于在大型的 Web 站点中，用户请求的次数肯定会很多，如果每一个请求都要响应一次数据库的连接与关闭，将会严重地消耗系统资源，甚至会导致系统崩溃。因此，使用数据库连接池技术是必要的，也是可取的。

## 12.3 Connection Pool 技术的原理

连接池最基本的思想就是预先建立一些连接，放置于内存对象中以备使用。当程序中需要建立数据库连接时，只需从内存中取一个来用而不用新建。同样，使用完毕后，只需放回内存即可。而连接的建立、断开都有连接池自身来管理。同时，我们还可以通过设置连接池的参数来控制连接池中的连接数、每个连接的最大使用次数等。

数据库连接池的实现，是由一个 `PoolManager` 类组成，该类管理 `ConnectionPool` 类的多个实例，每个 `ConnectionPool` 管理 `JDBC Connection` 对象的一个池。

所有的连接池客户都可以与 `PoolManager` 进行交互，也可以与 `Connection` 对象交互，但是不直接与 `ConnectionPool` 对象进行交互。`PoolManager` 保存了 `ConnectionPool` 对象的

一个集合，每个对象负责惟一的 JDBC URL 和数据库账户。PoolManager 的两个方法 getConnection()和 freeConnection()都有一个池名称属性，该属性由 PoolManager 用来转发请求到正确的 ConnectionPool 对象。

下面来讲述一下 ConnectionPool 类和 PoolManager 类。

### 12.3.1 ConnectionPool 类

ConnectionPool 类的方法主要用于：

- 从数据库池中获得一个打开的连接
- 返回一个连接到数据库池
- 释放资源并关闭所有的连接

一个 ConnectionPool 对象有一个客户所用的名称，该对象也表示一个数据库的连接池。用一个 JDBC URL 标识数据库，一个 JDBC URL 包含三部分：协议标识符、驱动程序标识符和数据库标识符。对于需要数据库账号才能连接的数据库，可以定义一个数据库用户名称和一个口令。

在实际的应用中，如果应用允许所有的用户访问某些数据库，但另外一些表只对授权用户访问，那么您就可以为普通用户定义一个连接池，同时为使用相同的 JDBC URL 而用户名和口令不同的授权用户单独定义一个连接池。

ConnectionPool 类定义了许多实例变量，这些实例变量用来保持客户任何地方都可以使用的 JDBC Connection 对象，同时还用来跟踪连接池中的连接总数。

ConnectionPool 构造函数除了保存实例变量的参数值以外，还用三个参数 String name、PrintWriter pw 和 int logLevel 创建了一个 LogWriter。

```
Public ConnectionPool(String name,String url,String user,String password,int maxConnections,
    int initConnection,int timeOut,PrintWriter pw,int logLevel)
{
    this.name=name;
    this.url=url;
    this.user=user;
    this.password=password;
    this.maxConnections = maxConnections;
    this.timeOut = timeOut > 0? timeOut:0;
    logWriter =new LogWriter(name, logLevel, pw);
    initPool(initConnection);
    logWriter.log("new connection pool is created",LogWriter.INFO);
    String ls=System.getProperty(line.separator);
    LogWriter.log(ls+" url= "+url+" user= "+user+" password= "+password+"
        "maxConnections= "+ maxConnections+" logintime= "+
        this.timeOut,LogWriter.DEBUG);
    LogWriter.log(getStatus(), logWriter.DEBUG());
}
```

ConnectionPool 构造函数在实例变量中保存大多数的参数值，并且调用 initPool()方法打开初始连接。InitPool()方法用于打开指定数目的连接，并且把它们增加到初始变量中。



InitPool()方法是一个从 ConnectionPool 构造函数中调用的 private 方法，代码如下所示：

```
private void initPool (int initValue)
{
    for(int i=0;i<initValue;i++)
    {
        try
        {
            Connection conn=newConnection();
            //vectorConn 是一个 Vector 类型的实例变量
            vectorConn.addElement(conn);
        }
        catch(Exception e)
        {
            System.err.println(e.getMessage());
        }
    }
}
```

值得注意的是当数据库引擎还没有运行时，会抛出异常。

ConnectionPool 类还提供了一个获得连接的 public 方法 getConnection()。该方法以指定的最大等待时间作为参数，返回值为 Connection。

```
public Connection getConnection ()
{
    try
    {
        return getConnection(time);
    }
    //如果不能返回一个 Connection，将会抛出异常
    catch(Exception e)
    {
        logWriter.log(e,"",LogWriter.ERROR);
    }
}
```

如果连接池中有可以使用的连接，就从连接池中返回一个 Connection；否则，如果连接池中的连接数还没有达到最大的连接数，就会建立一个新的连接。如果连接池中沒有可使用的 Connection 并且连接池中的连接数已经达到最大连接数，那么就必须等待由运行在不同线程中的其他用户返回连接池中的 Connection。

为了能够重新利用已经发送给用户的 Connection，在提供 Connection 时，就必须为用户提供一个返回 Connection 到连接池中的方法。ConnectionPool 有一个 freeConnection()方法可以将 Connection 返回到连接池中。其代码如下：

```
public synchronized void freeConnection(Connection conn)
{
    vectorConn.addElement(conn);
    logWrite.log("return to connection pool", LogWriter.INFO);
    logWrite.log(getStatus(), LogWriter.DEBUG);
}
```

```
}
```

最后, ConnectionPool 还提供了一个完美地释放所有池资源的方法。当最后一个用户不再使用连接池服务时, 由 PoolManager 调用 release()方法释放所有的池资源。

```
public void release()
{
    Enumeration allConnection= vectorConn.elements();
    while(allConnection.hasMoreElements())
    {
        Connection connection=(Connection) allConnection.nextElement();
        try
        {
            connection.close();
            logWriter.log("connection is closed.", logWriter.info);
        }
        catch(Exception e)
        {
            logWriter.log(e,"",LogWriter.ERROR);
        }
    }
    allConnection.removeAllElements();
}
```

### 12.3.2 PoolManager 类

一个 ConnectionPool 实例处理 Connection 对象的一个池, 这些 Connection 对象使用相同的 JDBC URL 和数据库用户帐户访问数据库引擎。PoolManager 提供了用于用户应用的接口, 能够使用多个连接池。

PoolManager 类提供了以下功能:

- 加载和注册所有的 JDBC 驱动程序。
- 根据一个特性文件中定义的特性, 创建 ConnectionPool 对象。
- 将连接池名称映射到 ConnectionPool 对象上。
- 将客户请求传递到一个名称为 ConnectionPool 的特定对象。
- 跟踪连接池中的用户, 并能够关闭连接池。

PoolManager 有两个静态变量 instance 和 clients。如:

```
static private PoolManager instance;
static private int clients;
```

其中, instance 变量用于保存 PoolManager 类的单个实例的引用, clients 变量用于记忆有多少个用户使用连接池。当用户调用 getInstance()方法时, clients 变量就会增加 1; 而当用户调用 release()方法时, clients 变量则会减小 1。当 clients 变量值为零时, 说明没有用户使用连接池, 那么就可以关闭所有的 ConnectionPool 对象了。

PoolManager 还定义了两个实例变量 drivers 和 pools。如:

```
private Vector drivers = new Vector();
private Hashtable pools = new Hashtable();
```

其中, `drivers` 变量用于跟踪 JDBC 驱动程序, `pools` 变量则用于跟踪 `ConnectionPool` 实例。

`PoolManager` 的构造函数是 `private` 的, 它能够防止其他对象创建该类的实例。

```
private PoolManager()
```

```
{  
    init();  
}
```

`PoolManager` 的静态方法 `getInstance()` 可以用于获得单个实例的引用。

```
static public PoolManager getInstance()
```

```
{  
    if(instance==null)  
    {  
        instance=new PoolManager();  
    }  
    clients++;  
    return instance;  
}
```

当第一次调用 `getInstance()` 方法时, 创建单个实例并且将该实例的引用保存在名称为 `instance` 的静态变量中。在将该引用返回之前, 将 `PoolManager` 用户数目加 1。

私有方法 `createPools()` 是用来创建 `ConnectionPool` 对象的, 如:

```
private void createPools(Properties prop)
```

```
{  
    Enumeration propNames= prop.propertyNames();  
    while(propNames.hasMoreElements())  
    {  
        String name=(String)propNames.nextElement();  
        if(name.endsWith(".url"))  
        {  
            String poolName=name.substring(0,name.lastIndexOf("."));  
            String url=prop.getProperty(poolName+".url");  
            if(url==null)  
            {  
                logWriter.log("no URL searched.",LogWriter.ERROR);  
                continue;  
            }  
        }  
    }  
}
```

`PoolManager` 有两个方法 `getConnection()` 和 `freeConnection()`。其中, `getConnection()` 方法是用于获得连接的, 而 `freeConnection()` 方法则是用于释放连接的。

方法 `getConnection()` 和 `freeConnection()` 都是使用连接池名称作为一个参数, 并调用转发到相应的 `ConnectionPool` 对象上的。如:

```
public Connection getConnection(String name)  
{
```

```
Connection conn;
conn=null;
ConnectionPool cp=(ConnectionPool)pools.getName();
if(pool!=null)
{
    try
    {
        conn=pool.getConnection();
    }
    catch(Exception e)
    {
        logWriter.log(e,"Exception",LogWriter.ERROR);
    }
}
return conn;
}
public void freeConnection(String name, Connection conn)
{
    ConnectionPool pool=(Connection)pools.getName();
    if(pool!=null)
    {
        pool.freeConnection(conn);
    }
}
```

最后，讲述一下 release()方法。

release()方法是用于关闭数据库连接池的。每个 PoolManager 用户调用静态 getInstance() 方法获得管理器的一个引用。当服务器关闭时，每个用户必须调用 release()方法，以便确保连接数据库连接池的数目减 1。当最后一个用户调用 release()方法时，PoolManager 对所有的 ConnectionPool 对象调用 release()方法以便于关闭所有的连接。当释放所有的 ConnectionPool 对象后，就会撤销所有的 JDBC 驱动程序的注册。

## 12.4 常用服务器的 Connection Pool 的配置工作

下面将讲述一下一些常用的服务器的数据库连接池的配置工作。

### 12.4.1 weblogic 服务器的配置

在 weblogic 的属性配置文件中，下面的配置是用来设置数据库的：

```
1      #weblogic.jdbc.connectionPool.wxnPool=
2      # url=jdbc:oracle:thin:@192.168.1.100:1221:wxn1,
3      # driver=oracle.jdbc.driver.OracleDriver,
4      # loginDelaySecs=1,
5      # initialCapacity=4,
```

```
6      # maxCapacity=10,  
7      # capacityIncrement=2,  
8      # allowShrinking=true,  
9      # shrinkPeriodMins=15,  
10     # refreshMinutes=10,  
11     # testTable=table,  
12     # props=user=system;password=manager;server=serverName
```

以下是上述配置的描述:

第一行: 设置 JDBC POOL 的名字(如 wxnPool);

第二行: 连接的数据库;

第三行: 注册 JDBC 驱动程序;

第五行: 默认的连接数;

第六行: 连接池中最大连接数;

第十二行: 连接数据库的用户名和密码。

默认情况下, 数据库连接池是不允许使用的。如果要使用 JDBC POOL 可以将前面的注释去掉, 前提是 JDBC 驱动程序必须是在 CLASSPATH 中已经存在。

### 12.4.2 resin 服务器的配置

在 resin 的属性配置文件中, 下面的配置是用来设置数据库的:

```
1  <dbpool.sql>  
2  <id>ORCL</id>  
3  <driver>oracle.jdbc.driver.OracleDriver</driver>  
4  <url>jdbc:oracle:thin:@localhost:1521:SMTH</url>  
5  <user>scott</user>  
6  <password>tiger</password>  
7  <max-connections>5</max-connections>  
8  </dbpool.sql>
```

以下是上述配置的描述:

第二行 id: DBPool 系统中的表示名称;

第三行 driver: JDBC 的 driver, 与所使用的数据库有关;

第四行 url: 数据库的 url 表示, 也与所使用的数据库有关;

第五行 user: 数据库用户名;

第六行 password: 数据库密码;

第七行 max-connections: 最大连接池连接数目。

## 12.5 Connection Pool 使用实例

下面讲述在 Servlet 中使用连接池的例子。

Servlet API 所定义的 Servlet 生命周期分三部分:

- 创建并初始化 Servlet (init()方法)

- 响应客户程序的服务请求（service()方法）
- Servlet 终止运行，并释放所有资源（destroy()方法）

在连接池的应用中，上述关键步骤中的相关操作分别为：

- 在 init()，用实例变量 connMgr 保存调用 DBConnectionManager.getInstance()所返回的引用。
- 在 service()，调用 getConnection()，执行数据库操作，用 freeConnection()将连接返回给连接池。
- 在 destroy()，调用 release()关闭所有连接，释放所有资源。

范例程序代码如例程 12-1 所示。

例程 12-1

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class connPool extends HttpServlet
{
    private DBConnectionManager connMgr;

    public void init(ServletConfig conf) throws ServletException
    {
        super.init(conf);
        connMgr = DBConnectionManager.getInstance();
    }

    public void service(HttpServletRequest req, HttpServletResponse res)
        throws IOException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        Connection con = connMgr.getConnection("dbname");
        if (con == null) {
            out.println("database connection failed.");
            return;
        }
        ResultSet rs = null;
        ResultSetMetaData md = null;
        Statement stmt = null;
        try {
            stmt = con.createStatement();
            rs = stmt.executeQuery("SELECT * FROM member");
            md = rs.getMetaData();
            out.println("<H1>member</H1>");
            while (rs.next()) {
```



```
out.println("<BR>");
for (int i = 1; i < md.getColumnCount(); i++) {
    out.print(rs.getString(i) + ", ");
}
}
stmt.close();
rs.close();
}
catch (SQLException e) {
    e.printStackTrace(out);
}
connMgr.freeConnection("dbname", con);
}

public void destroy() {
    connMgr.release();
    super.destroy();
}
}
```

---

# 开发专家之

# un ONE

## 第五篇 基于 XML 的 JSP 应用

在本篇中，将讲述在基于 XML 的 JSP 应用。

本篇分为一章，讲述了 JSP 的 XML 语法、JSP 与 XML 文档的映射以及在 JSP 中如何使用 XML 数据源等。

通过对本篇的学习，将会掌握在 JSP 开发中是如何结合 XML 开发及充分利用 XML 优点的。



## 第 13 章 JSP 与 XML 文档

在本章中，讲述了基于 XML 的 JSP 应用。首先讲述了 JSP 的 XML 语法，包括 `jsp:root` 元素、`jsp:directive.page` 元素、`jsp:directive.include` 元素、`jsp:declaration` 元素、`jsp:scriptlet` 元素、`jsp:expression` 元素、`jsp:cdata` 元素等。

在第二节中讲述了 JSP 与 XML 文档的映射，包括 `page` 编译指令、`taglib` 编译指令、`include` 编译指令、变量声明指令、Scriptlets 指令、表达式等。

最后则介绍了在 JSP 中如何使用 XML 数据源，包括将 XML 转换成服务器端对象并提取对象属性、用 XSLT 转换 XML 以及使用 JSP 生成文档等。

通过对本章的学习，将会掌握在 JSP 开发中是如何结合 XML 开发的并充分利用 XML 优点的。

### 13.1 JSP 的 XML 语法

所有的 JSP 页面都有一个等价的 XML 文档，当然也可以直接书写与 JSP 等价的 XML 文档形式。在介绍 JSP 页面的 XML 语法之前，先了解几个 XML 的基本概念。

XML 是“Extensible Markup Language”的缩写，其意思是“可扩展标记语言”。XML 是 SGML 的一个子集，其目标是能够以目前 HTML 可能实现的方式在 Web 上使用、接受和处理通用的 SGML。XML 的设计目标是实现简便，并且能与 SGML 和 HTML 共同操作。

XML 最大的好处之一是它为用户提供了创建文档的灵活性。然而，为了最充分的利用这种灵活性，创建文档时应考虑使用的文档类型定义（DTD）。

DTD 是“Document Type Declaration”的缩写，其意思是“文档类型定义”。DTD 为 XML 文档定义应该包含或者可以包含的元素。

理解了上述几个基本的概念之后，下面讲述以下 JSP 页面的 XML 语法。

#### 13.1.1 jsp:root 元素

JSP 文档用 `jsp:root` 作为自身的根元素类型。Root 是 `taglib` 插入它们名称属性的地方。顶层元素有 `xmlns` 属性，因此可以使用 JSP1.2 规范中定义的标准元素。

所有的 JSP 文档中使用的标记库都表示为根元素的附加 `xmlns` 属性。这个元素中没有其他属性。

其表现形式为：

```
<jsp:root
  xmlns:jsp="http://java.sun.com/jsp_1_2"
  xmlns:prefix1="URI1"
  xmlns:prefix1="URI1"....>
```

```
</jsp:root>
```

自定义标记库的 `xmlns` 属性的形式 `xml:prefix="URI"` 通过 `URI` 的值表示标记库, `URI` 有形式 `"urn:jsp:tld:path"`, 描述怎样为 JSP 语法形式的 JSP 页面中的 `taglib` 指令定位标记库描述符。

### 13.1.2 jsp:directive.page 元素

`page` 指令是用来定义 JSP 文件中的全局变量的。`<%@ page attribute="value" %>` 作用于 Servlet 引擎的全局性指令。

等价的 XML 表达是:

```
<jsp:directive.page attribute="value"/>。
```

合法的属性如下所示, 其中粗体表示默认值。

```
language="java"
import="package.class"
contentType="MIME-Type"
isThreadSafe="truefalse"
Session="truefalse"
buffer="8kb| size kb|none"
autoFlush="truefalse"
extends="package.class"
info="message"
errorPage="url"
isErrorPage="truefalse"
```

### 13.1.3 jsp:directive.include 元素

在 JSP 中, `include` 指令 `<%@ include file="url" %>` 包含一个静态的文件, 同时解析这个文件中的 JSP 语句。当 JSP 转换成 Servlet 时, 应当包含本地系统上的指定文件。

等价的 XML 表达是:

```
<jsp:directive.include file="url"/>。
```

其中 URL 必须是相对 URL。

### 13.1.4 jsp:declaration 元素

在 JSP 声明中, `<%! code %>` 代码被插入到 Servlet 类 (在 `Service` 方法之外)。等价的 XML 表达是:

```
<jsp:declaration>
code
</jsp:declaration>
```

### 13.1.5 jsp:scriptlet 元素

在 JSP 程序段（Scriptlet）中，`<% code %>` 代码被插入到 Service 方法中。  
等价的 XML 表达是：

```
<jsp:scriptlet>
code
</jsp:scriptlet>
```

### 13.1.6 jsp:expression 元素

在 JSP 表达式中，`<%= expression %>` 计算表达式 expression 并输出结果。  
等价的 XML 表达是：

```
<jsp:expression>
expression
</jsp:expression>
```

### 13.1.7 jsp:cdata 元素

jsp:cdata 元素用来封装 XML 中的 template data。jsp:cdata 元素没有属性值并且能够出现在 template data 出现的任何地方。而 template data 只能出现在属性或 jsp:cdata 元素中。

基本表达式是：

```
<jsp:expression>
cdata
</jsp:expression>
```

jsp:cdata 元素的解释是把它的内容传递给 out 的当前值。

## 13.2 JSP 与 XML 文档的映射

### 13.2.1 page 编译指令

page 指令定义 JSP 文件中的全局属性。

其语法为：

```
<%@ page
[ language="java" ]
[ import="{package. class | package.*}, ..." ]
[ contentType="TYPE; charset=CHARSET" ]
[ Session="true | false" ]
[ buffer="none | 8kb | sizekb" ]
[ autoFlush="true | false" ]
[ isThreadSafe="true | false" ]
```



```
{ info="text" ]  
[ errorPage="relativeURL" ]  
[ isErrorPage="true | false" ]  
{extends="package. class" ]  
%>
```

属性为:

1. language="java"

声明脚本语言的种类, 默认情况下为“java”。可能的其他值为 java、javascript 和 web1。

2. import="{package.class | package.\* }, ..."

需要导入的 Java 包的列表, 这些包就作用于程序段, 表达式, 以及声明。

下面的包在 JSP 编译时已经导入了, 所以就不需要再指明了:

```
java.lang.*  
javax.servlet.*  
javax.servlet.jsp.*  
javax.servlet.http.*
```

3. contentType ="TYPE; charset=CHARSET"

设置 MIME 类型。默认 MIME 类型是: text/html, 默认字符集为 ISO-8859-1。

语法如下:

```
<% page contentType ="TYPE; charset=CHARSET" %>
```

4. Session="true | false"

设定客户是否需要 HTTP Session。

如果它为 true, 那么 Session 是有用的。

如果它为 false, 那么就不能使用 Session 对象, 以及定义了 scope=Session 的 <jsp:useBean>元素。这样的使用会导致错误。

默认值是 true。

5. buffer="none | 8kb | sizekb"

buffer 的大小被 out 对象用于缓存处理执行后的 JSP 对客户浏览器的输出。

“none”是指没有任何缓存, 直接输出到客户端浏览器。

您可以通过制定 buffer 的大小来制定缓存处理的大小。

默认值是 8KB。

6. autoFlush="true | false"

用来设置当 buffer 溢出时, 是否需要强制输出。如果其值被定义为 true, 则输出正常; 如果它被设置为 false, 当 buffer 溢出时, 就会导致一个意外错误的发生, 如果把 buffer 设置为 none, 那么就不能把 autoFlush 设置为 false。

默认值是 true。

7. isThreadSafe="true | false"

用来设置 Jsp 文件是否能多线程使用。

如果设置为 true, 那么一个 JSP 能够同时处理多个用户的请求; 相反, 如果设置为 false, 一个 JSP 只能一次处理一个请求。

默认值是 true。

## 8. info="text"

在 JSP 被执行时，用来描述当前 JSP 文件的相关信息。您可以通过使用 `Servlet.getServletInfo()` 方法取得。

## 9. errorPage="relativeURL"

设置处理异常事件的 JSP 文件。

需要注意的是如果您设置 `autoFlush` 为 `true`，当该页的输出产生意外时，可能会导致失败。

## 10. isErrorPage="true | false"

设置此页是否为出错页。

如果被设置为 `true`，则可以使用 `exception` 对象。相反，如果被设置为 `false`，则不可以使用 `exception` 对象。

默认值是 `false`。

## 11. extends="package.class"

标明 JSP 编译时需要加入的 Java Class 的全名，但是得慎重地使用它，它会限制 JSP 的编译能力。

定义 XML 指令的语法是：

```
<jsp:directive.page attribute=value />
```

例如：

```
<%@ page import="java.util.*" %>
```

相当于

```
<jsp:directive.page import="java.util.*" />
```

### 13.2.2 taglib 编译指令

JSP 中定义了 Taglib 标记，应用这个标记可以实现 XML 与 JSP 的有机结合。

Taglib 指令：定义一个标签库以及其自定义标签的前缀。

语法为：

```
<%@ taglib uri="URIToTagLibrary" prefix="tagPrefix" %>
```

语法描述：

Taglib 指令声明了 JSP 文件使用了自定义的标签，同时引用标签库，也指定了它们的标签的前缀。

属性：

## 1. uri="URIToTagLibrary"

Uniform Resource Identifier (URI) 根据标签的前缀对自定义的标签进行唯一的命名，URI 可以是以下内容：

Uniform Resource Locator (URL)；

Uniform Resource Name (URN)

## 2. prefix="tagPrefix"

在自定义标签之前的前缀与自定义标签中的元素是一致的。

例如：

```
<%@ taglib uri="http://sun.java.com/taglib" prefix="simple" %>
<simple:name>
...
</simple:name>
```

### 13.2.3 include 编译指令

在 JSP 文件中用 Include 指令包含一个静态的文件，同时解析这个文件中的 JSP 语句。  
基本语法：

```
<%@ include file="path" %>
```

包含文件的路径名一般来说是指相对路径，不需要什么端口、协议和域名。如下所示。

```
<%@ include file="header.inc"%>
```

如果路径以 "/" 开头，那么路径主要是参照 JSP 应用的上下关系路径；如果路径是以文件名或目录名开头，那么这个路径就是正在使用的 JSP 文件的当前路径。

等价地，XML 可以用以下表达式来表示：

```
<jsp:directive.include file="url"/>.
```

其中 URL 必须是相对 URL。

例如：

```
<%@ include file=" error.html " %>
```

相当于

```
<jsp:directive.include file="error.html"/>
```

### 13.2.4 变量声明指令

在 JSP 程序中，声明用于声明合法的变量和方法。表现形式如下：

```
<%! 声明 %>
```

例如：

```
<%! String user_name; %>
<%! int a,b,c; %>
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
```

```
}
%>
```

等价地，XML 可以用以下表达式来表示：

```
<jsp:declaration>
  声明
```

```
</jsp:declaration>
```

相应地，例子代码为：

```
<jsp:declaration>
  String  user_name;
</jsp:declaration>
<jsp:declaration>
  int  a,b,c;
</jsp:declaration>
<jsp:declaration>
  public String getStr(String str)
  {
    try
    {
      String temp_p=str;
      byte[] temp_t=temp_p.getBytes("GBK");
      String temp=new String(temp_t,"ISO8859_1");
      return temp;
    }
    catch(Exception e)
    {

    }
    return "null";
  }
</jsp:declaration>
```

### 13.2.5 Scriptlets 指令

Scriptlets 指令能够将任意 Java 代码插入到 Servlet 方法中，最终产生理想的网页。其表现形式如下：

```
<% 程序代码 %>
```

Scriptlets 指令和表达式一样可以利用预定义变量。因此，如果要输出结果，则可以应用 out 变量。

```
<%
String  now_time = new java.util.Date();
out.println("当前时间为 " + now_time);
%>
```

脚本内部的代码是被准确写出的（就是 Servlet 方法），而它之前或之后的任何静态文本（模本文本）则需要转换成输出流。这就意味这脚本不一定必须要完整的 Java 表达式，

而可以通过静态文本的开放式块输出。例如下面这段程序，就是静态模板文本与脚本的混合体。

```
<%
    String name;
    if(request.getParameter("name")==null)
    {
%>
您没有输入用户名!
    <%
    }
else
    {
    name= request.getParameter("name");
%>
    您好, <% =name %>, 欢迎光临。
    <%
    }
%>
```

经过转译后将会是如下的形式:

```
<%
String name;
if(request.getParameter("name")==null)
{
    out.println("您没有输入用户名! ");
}
else
{
    out.println("您好, "+name +", 欢迎光临。");
}
%>
```

等价地, XML 可以用以下表达式:

```
<jsp:scriptlet>
程序代码
</jsp:scriptlet>
```

相应地, 例子代码为:

```
<jsp:scriptlet>
String name;
if(request.getParameter("name")==null)
{
    out.println("您没有输入用户名! ");
}
else
{
    out.println("您好, "+name +", 欢迎光临。");
}
```

```
</jsp:scriptlet>
```

### 13.2.6 表达式

表达式用来直接输出 Java 的expressions的值，表现形式如下：

```
<%= Java 表达式 %>
```

Java 表达式被计算出来，转换成字符串形，然后输出到网页中。expressions的值是在运行过程中计算出来的，因此能直接与网页的请求相关联。

例如，下面的代码要求网页输出当时的时间。

当前时间：<%= new java.util.Date() %>

为了简化这些expressions，有许多预定义变量可以利用。这些对象已经在前面的章节中讲述并且也大量地使用过。

下面是一个例子：

您的主机名是：<%= request.getRemoteHost() %>

同样，XML 可以用以下expressions表达：

```
<jsp:expression>  
Java 表达式  
</jsp:expression>
```

记住 XML 是大小写敏感的，一定要注意用小写。

例子代码为：

当前时间：

```
<jsp:expression>  
new java.util.Date()  
</jsp:expression>
```

以及

您的主机名是：

```
<jsp:expression>  
request.getRemoteHost()  
</jsp:expression>
```

## 13.3 确认 JSP 文档

JSP 文档是一个名域确定的文档。然而除了在相当简单的场合，否则它不能使用 DTD 来确认。当然，DTD 描述文档的目的是有用的。确定名域的机制，如 Xschema，相当适合描述 JSP 文档。

## 13.4 在 JSP 中使用 XML 数据源

在 JSP 页面中可以使用多种数据源，如 XML 数据源、JDBC 数据源、EJB 数据源。一个页面通过服务器端对象与数据源相连，将信息转换成数据抽象，接着用 JSP 元素显示



数据。本节主要来讲述如何在 JSP 页面中使用 XML 数据源。

假设已经存在如例程 13-1 所示的一个 XML 程序，该 XML 程序表示用户的个人资料。

例程 13-1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
  <user ID="111">
    <userName>wang xining</userName>
    <PassWord>*****</PassWord>
    <sex>male</sex>
    <age>22</age>
  </user>
  <user ID="222">
    <userName>Blue River</userName>
    <PassWord>*****</PassWord>
    <sex>male</sex>
    <age>88</age>
  </user>
  <user ID="333">
    <userName>Red Sun</userName>xxxx
    <PassWord>*****</PassWord>
    <sex>female</sex>
    <age>66</age>
  </user>
</users>
```

有两种方法可以在 JSP 页面中使用这个 XML 数据：

- 将 XML 元素转换成服务器端对象，然后从对象属性中提取数据。
- 直接激活一个 XML 数据转换程序。

### 13.4.1 将 XML 元素转换成服务器端对象并提取数据

在这种方法中，需要创建一些代表 XML 文档所承载的信息的对象。目前，必须写代码来完成这个功能，使用 SAX 或 DOM，并将这些对象封装成自定义标签或 JavaBeans 组件。

SAX 和 DOM 是两个基本的解析模型。

SAX 基于事件，所以在 XML 被解析时，事件被发送给引擎。接下来，事件与输出文件同步。DOM 解析引擎为动态 XML 数据和 XSL 样式表建立层次树状结构。通过随机访问 DOM 树，可以提供 XML 数据，就像由 XSL 样式表来决定一样。SAX 模型上的争论主要集中于对 DOM 结构的内存降低过度和加快 XSL 样式表解析时间缩短方面。

将来 XML/Java 的绑定技术将自动将一个 XML 模式编译成 Java 类完成处理。

一个 JSP 页面可以从 URL 中检索 XML 数据，并且使用这些数据生成一个 HTML 页面，这个 URL 可以指向一个能动态生成 XML 数据的地方或者是一个静态的 XML 文档，

也可以用同样的技术生成 XML。

JSP 页面使用解析器解析自定义标签，并在对象中存储 XML 数据。然后，这个页面从对象中抽取属性值，从对象属性中提取数据。

### 13.4.2 用 XSLT 转换 XML

另一种在 JSP 页面中使用 XML 数据源的方法是在 XML 数据源上使用一个转换程序，或者是抽取数据或者是创建新的格式。这个转换程序可以使用许多不同的机制来实现并且通过自定义标签来访问它。

XSLT (eXtensible Stylesheet Language for Transformations 可扩展样式表变换语言) 是 W3C 小组制定的一个转换语言规范，它可以用来将 XML 数据转换成 HTML、PDF 或其他 XML 格式。

为了使用这个方法生成 HTML 页面，需要一个 XSL 样式页和一个应用这个样式页的方法。

### 13.4.3 使用 JSP 生成文档

近年来，随着网络的发展和普及，基于 Web 的服务得到了广泛的使用，并且将有越来越多类型的用户访问 Web 服务。目前，不同种类面向用户的基于 Web 应用程序的用户主要有：PDA、WAP 移动电话和 landline Voice 用户。随着 B2B 交易的增长，使用 XML 的服务器也可以成为一个用户端。

这些用户使用如下几种语言，如表 13-1 所示。

表 13-1 XML 语言分类对比

用 户	标 记 语 言
基于 PC 机的浏览器	HTML、DHTML、XHTML
PDA (Personal Digital Assistant, 个人数字助理)	WML、XHTML
移动电话	WML、XHTML
LandLine 电话	VoiceXML
服务器	特定应用程序的 XML 语言

支持上述所有的这些不同类型的用户，对于 HTML 的应用程序来说是不能够实现的。要想同时满足上述不同种类的用户，建议采用 JSP 和 XML 技术。

Web 应用程序开发人员传统上使用 JSP 技术动态构建 HTML，方法是将 Java 代码包括在 HTML 源代码中。但是也可以使用同样的方法生成 HTML 之外的动态内容，而且方法比较简单。

可以使用 XML 文档构建 JSP 页面，该 XML 文档将用作输出模板，然后替换必须基于基层业务逻辑动态生成的部分。为了生成文档的动态部分，您既可以使用直接编写在 JSP 页面中的 Java 代码，也可以使用从该页面外部调用的 Java 代码。

生成文档的哪些部分是由程序员自己控制的呢？例如，既可以使用 Java 代码生成两

个 XML 标记之间的数据，又可以生成 XML 文档树的各个部分（标记和数据），甚至可以生成整个文档。

Java 代码被从页面中除去，并被加工成一个 Servlet（称为页面 Servlet），然后 Java 应用程序服务器将其作为 JSP 页面请求的一部分运行。得到的结果是纯 XML。

通过使用 JSP 页面，页面内的静态 XML 就可以充当一个模板，该模板是用动态内容填充的。Java 代码的任务仅仅是生成可以随时间变化的内容——这是一种更有效的方法。

一个 JSP 页面可以生成一个包含文档的响应，生成 XML 的主要要求是 JSP 页面可以恰当地设置页面的内容类型。如下代码所示。

```
<%@ page contentType="text/xml"%>
<%
    out.println("<?xml version='1.0' encoding='ISO-8859-1'?>");
    out.println("<users>");
    out.println("    <user ID='111'>");
    out.println("        <userName>wang xining</userName>");
    out.println("        <PassWord>*****</PassWord>");
    out.println("        <sex>male</sex>");
    out.println("        <age>22</age>");
    out.println("    </user>");
    out.println("    <user ID='222'>");
    out.println("        <userName>Blue River</userName>");
    out.println("        <PassWord>*****</PassWord>");
    out.println("        <sex>male</sex>");
    out.println("        <age>88</age>");
    out.println("    </user>");
    out.println("    <user ID='333'>");
    out.println("        <userName>Red Sun</userName>xxxx");
    out.println("        <PassWord>*****</PassWord>");
    out.println("        <sex>female</sex>");
    out.println("        <age>66</age>");
    out.println("    </user>");
    out.println("</users>");
%>
```

经过这个变动，先前所讨论生成 HTML 内容的技术就可以用来生成 XML 了。

# 开发专家之

# Sun ONE

## 第六篇 JSP 实际开发中的应用

在本篇中，将讲述 JSP 技术在实际开发中的应用。

本篇分为九章，包括网站规划和设计、用户注册登录、查询商品、商品分类、最新、特价及缺货商品列表、购物车、用户订单、论坛、聊天室等。

通过对本篇的学习，将会掌握如何在实际的应用中，使用 JSP 技术进行开发。





# 第 14 章 网站规划和设计

下面将讲述如何使用 JSP 技术构建网站。在这里我们要创建的站点是一个在线超市的系统，由于我们的目的是教会读者如何使用 JSP 技术，所以本站点简单清晰。其实，目前真正的大型网站也就是我们这个小站点在内容上地扩充和重复。

以下我们将讲述整个网站的整体规划和设计。

## 14.1 网站的整体结构和站点的创建

在 Tomcat 的设置中，创建自己的新目录。如下代码所示，我们就在 webapps 目录下创建了新目录 test。这样，除了原有的 examples 和 Root，又多了两个目录。

```
<Context path="/test" docBase="webapps/test" debug="0" reloadable="true" >
</Context>
```

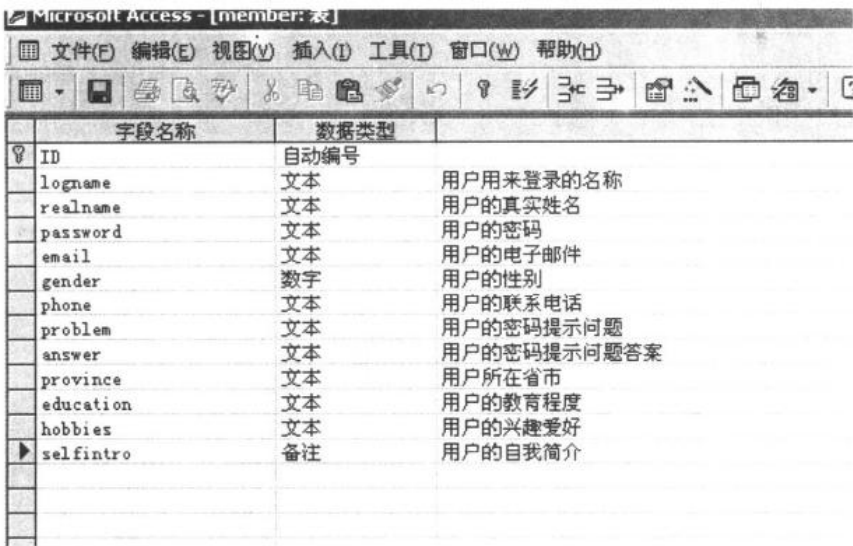
我们的站点在线超市电子商务系统是 test 目录下的 ec 目录。这样该站点的地址就是：  
`http://localhost:9090/test/ec/`

## 14.2 建立数据库

对于该站点，需要建立一个含有三个表的数据库。下面对所用到的表逐一介绍。

### 1. 表 member

用来记录注册用户的个人资料，表结构如图 14-1 所示。



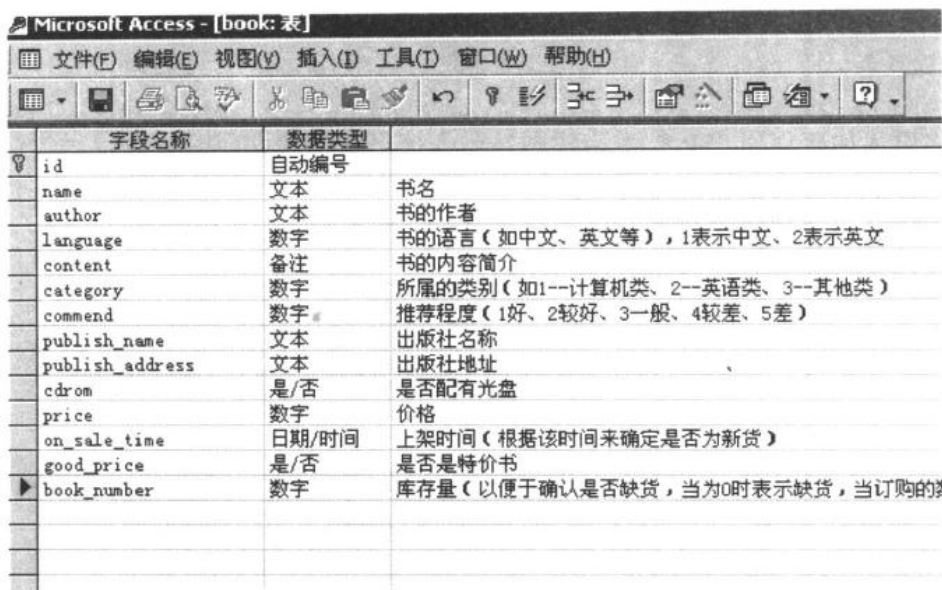
字段名称	数据类型	
ID	自动编号	
logname	文本	用户用来登录的名称
realname	文本	用户的真实姓名
password	文本	用户的密码
email	文本	用户的电子邮件
gender	数字	用户的性别
phone	文本	用户的联系电话
problem	文本	用户的密码提示问题
answer	文本	用户的密码提示问题答案
province	文本	用户所在省市
education	文本	用户的教育程度
hobbies	文本	用户的兴趣爱好
selfintro	备注	用户的自我简介

图 14-1 表 member



## 2. 表 book

用来记录商品（该例子指的是书）的信息，表结构如图 14-2 所示。



字段名称	数据类型	
id	自动编号	
name	文本	书名
author	文本	书的作者
language	数字	书的语言（如中文、英文等），1表示中文、2表示英文
content	备注	书的内容简介
category	数字	所属的类别（如1--计算机类、2--英语类、3--其他类）
commend	数字	推荐程度（1好、2较好、3一般、4较差、5差）
publish_name	文本	出版社名称
publish_address	文本	出版社地址
cdrom	是/否	是否配有光盘
price	数字	价格
on_sale_time	日期/时间	上架时间（根据该时间来确定是否为新货）
good_price	是/否	是否是特价书
book_number	数字	库存量（以便于确认是否缺货，当为0时表示缺货，当订购的

图 14-2 表 book

## 3. 表 orders

用来记录订单的信息，表结构如图 14-3 所示。



字段名称	数据类型	
	自动编号	
user_name	文本	订单人的姓名，必填项
user_address	文本	订单人的地址（以便于送货），必填项
user_tel	文本	订单人的电话（以便于送货），必填项
user_cid	数字	订单人的身份证号（以便于确认），必填项
book_id	数字	订购的书对应的id号，也就是表book中的id项
book_number	数字	订购的书的数量，至少为一本，当订购的数量超过库
status	数字	订单是否被处理（0表示未处理，1表示处理过）
goods_price	数字	货物的单价格

图 14.3 表 orders

## 14.3 公用的页面

为了使整个程序一致，建议使多次出现的部分代码封装在几个公用的页面中。例如：将每个页面的顶部和底部包含在 header.inc 和 footer.inc 文件中。

该站点用到了以下几个包含文件。

## 1. header.inc

这个文件放在每个页面的开头处，作为页面的顶部导航栏。代码如例程 14-1 所示。

例程 14-1

```
<HTML><HEAD><TITLE></TITLE>
<META content="text/html; charset=gb2312" http-equiv=Content-Type>
<LINK href="css/site.css" rel=stylesheet>
<META content="MSHTML 5.00.2014.210" name=GENERATOR></HEAD>
<script language="javascript" src="../js/all.js">
</script>
<BODY bgColor=#ffffff leftMargin=0 topMargin=5 marginheight="5" marginwidth="5">
  <table align="center" border="0" width="760" height="18" bgcolor="#3399FF" cellspacing="0">
    <tr>
      <td width="100%">
        <p><a class="x" href="member/list_all_user.jsp">用户列表</a> |
          <a class="x" href="book_store/index.jsp">商品列表</a> |
          <a class="x" href="../search/index.jsp">查询商品</a> |
          <a class="x" href="book_store/all_category.jsp">商品分类</a> |
          <a class="x" href="book_store/new_goods.jsp">新货上架</a> |
          <a class="x" href="book_store/good_price.jsp">特价市场</a> |
          <a class="x" href="book_store/short_goods.jsp">缺货登记</a> |
          <a class="x" href="book_store/shopcart.jsp" target="_blank">购物车</a> |
          <a class="x" href="book_store/order.jsp" target="_blank">订单</a> |
          <a class="x" href="forum/index.jsp" target="_blank">论坛</a> |
          <a class="x" href="chat/index.jsp" target="_blank">聊天室</a>
        </td>
      </tr>
    </table>
```

## 2. footer.inc

这个文件放在每个页面的结尾处，作为页面的底部信息。代码如例程 14-2 所示。

例程 14-2

```
<hr align="center" color="#3399ff" width="760" >
<table align="center" border="0" width="760">
  <tr>
    <td width="100%" height="18" align="center">
      <P class="foot">&copy; 2001 联系: <a href="mailto:blueriver_cn@sina.com">Blueriver</a>
    </P>
    <P class="foot">本系统随《JSP 应用开发详解》附送
    </P>
    </td>
  </tr>
</table>
```

```

</table>
</body>
</html>

```

### 3. table.inc

这个文件包含了每个页面的共同信息——注册用户的个数。代码如例程 14-3 所示。

例程 14-3

```

<table border="1" width="100%" bordercolorlight="#3399FF"
bordercolordark="#3399FF" cellspacing="0" cellpadding="0">
<tr>
<td width="100%" valign="top" style="border: 1 solid #3399FF">
<table border="0" width="100%" cellspacing="0">
<tr>
<td width="100%" bgcolor="#3399FF" height="18" align="center">
<font color="#ffffff">成员服务</font></td>
</tr>
<tr>
<td width="100%" height="23" align="center">
<a href="member/login.jsp"><b>登录本站</b></a>
<a href="member/reg.jsp"><b>成员注册</b></a></td>
</tr>
<tr>
<td height="18">
<tr>
<td width="100%" bgcolor="#3399FF" height="18" align="center">
<font color="#ffffff">本站统计</font></td>
</tr>
<tr>
<td width="100%">
<table border="0" width="98%" align="center" cellspacing="0">
<tr>
<td align="right" height="20" width="60%">注册用户: </td>
<%
int t_member; t_member=0;
String strSQLmem="SELECT id FROM member";
ResultSet RSmem = workM.executeQuery(strSQLmem);
while(RSmem.next()){
t_member=t_member+1;
}
%>
<td align="left" width="40%"><%=t_member%></td>
</tr>
</table>
</td>
</tr>
</table>

```


4. `date.inc`

这个文件包含了每个页面用到的时间。代码如例程 14-4 所示。

### 例程 14-4

[illegible]

## 14.4 JavaBeans

在该实例中，用到了以下几个 **Bean** 文件，下面将分别加以介绍。

## 1. 封装数据库

该实例采用了通过 JavaBeans 封装数据库连接。

源代码如例程 14-5 所示。

### 例程 14-5

```
package test;

import java.sql.*;

public class faq {

//JDBC-ODBC 桥

String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
String sConnStr = "jdbc:odbc:faq";
Connection conn = null;
ResultSet rs = null;

public faq() {
    try {
        Class.forName(sDBDriver);
    }
    catch(java.lang.ClassNotFoundException e) {
```

```
System.err.println(e.getMessage());
}
}
//定义 public 型函数
public ResultSet executeQuery(String sql) {
    rs = null;
    try {
        conn = DriverManager.getConnection(sConnStr);
        Statement stmt = conn.createStatement();
        rs = stmt.executeQuery(sql);
    }
    catch(SQLException ex)
    System.err.println("aq.executeQuery: " + ex.getMessage());
    }
    return rs;
}
}
```

## 2. 封装时间的 Bean 文件

通过这个 Bean 文件，可以不同形式地显示时间。

源代码如例程 14-6 所示。

例程 14-6

```
package dates;
import java.text.DateFormat;
import java.util.*;

public class JspCalendar {
    Calendar calendar = null;
    //构造函数
    public JspCalendar() {
        calendar = Calendar.getInstance();
        Date trialTime = new Date();
        calendar.setTime(trialTime);
    }
    //返回年
    public int getYear() {
        return calendar.get(Calendar.YEAR);
    }
    //返回月
    public String getMonth() {
        int m = getMonthInt();
        String[] months = new String [] { "1", "2", "3",
            "4", "5", "6",
            "7", "8", "9",
            "10", "11", "12" };
    }
}
```

```
        if (m > 12)
            return "Unknown to Man";

        return months[m - 1];
    }
    //返回星期几
    public String getDay() {
        int x = getDayOfWeek();
        String[] days = new String[] { "1", "2", "3",
                                         "4", "5", "6", "7" };

        if (x > 7)
            return "Unknown to Man";
        return days[x - 1];
    }
    public int getMonthInt() {
        return 1 + calendar.get(Calendar.MONTH);
    }
    //返回一月/日/年
    public String getDate() {
        return getMonthInt() + "/" + getDayOfMonth() + "/" + getYear();
    }
    //当前时间：时：分：秒
    public String getTime() {
        return getHour() + ":" + getMinute() + ":" + getSecond();
    }
    //返回当前时间是一月中的哪一天
    public int getDayOfMonth() {
        return calendar.get(Calendar.DAY_OF_MONTH);
    }
    //返回当前时间是一年中的哪一天
    public int getDayOfYear() {
        return calendar.get(Calendar.DAY_OF_YEAR);
    }
    //返回当前时间是一年中的哪个星期
    public int getWeekOfYear() {
        return calendar.get(Calendar.WEEK_OF_YEAR);
    }
    //返回当前时间是一年中的哪个星期
    public int getWeekOfMonth() {
        return calendar.get(Calendar.WEEK_OF_MONTH);
    }
    //返回当前时间是一周中的哪一天
    public int getDayOfWeek() {
        return calendar.get(Calendar.DAY_OF_WEEK)-1;
    }
}
```



```
//返回小时
public int getHour() {
    return calendar.get(Calendar.HOUR_OF_DAY);
}
//返回分钟
public int getMinute() {
    return calendar.get(Calendar.MINUTE);
}
//返回秒
public int getSecond() {
    return calendar.get(Calendar.SECOND);
}
public static void main(String args[]) {
    JspCalendar db = new JspCalendar();
    p("date: " + db.getDayOfMonth());
    p("year: " + db.getYear());
    p("month: " + db.getMonth());
    p("time: " + db.getTime());
    p("date: " + db.getDate());
    p("Day: " + db.getDay());
    p("DayOfYear: " + db.getDayOfYear());
    p("WeekOfYear: " + db.getWeekOfYear());
    p("era: " + db.getEra());
    p("ampm: " + db.getAMPM());
    p("DST: " + db.getDSTOffset());
    p("ZONE Offset: " + db.getZoneOffset());
    p("TIMEZONE: " + db.getUSTimeZone());
}
private static void p(String x) {
    System.out.println(x);
}
}
```

# 第 15 章 用户注册登录

本章将讲述 JSP 技术在用户注册登录表中的应用。在这里我们通过此实例，来讲述网站中最常用的技术。本章分为两节，第一节是用户注册系统，第二节是用户登录系统。下面分别进行介绍。

## 15.1 用户注册系统

本系统采用的用户注册方式是这样的：

首先，用户在页面 `reg.jsp` 中，输入必须填写的信息；并单击【提交】按钮来提交个人信息。

提交后，在 `reg_ok.jsp` 页面中进行验证。如果输入的用户名已经被其他用户注册使用过，那么系统会提示用户相应的信息。

如果输入的用户名没有被其他用户注册使用，并且系统要求必须填写的信息都合理，那么该用户就会成功注册。

由于本用户注册系统是独立的，因此完全可以把这个用户注册系统作以独立的子系统进行使用。

### 1. 页面注册效果图及其源代码

页面注册效果如图 15-1 所示。

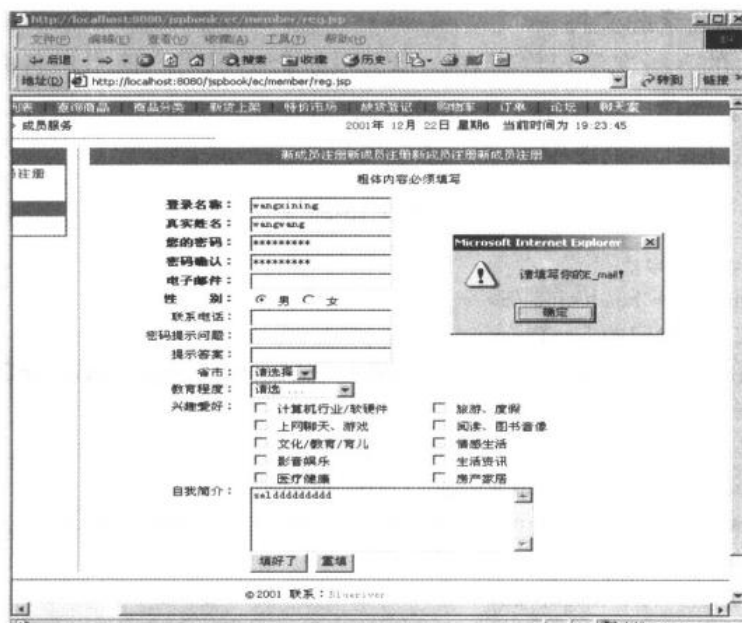


图 15-1 注册页面

文件 reg.jsp 的源代码如例程 15-1 所示。

例程 15-1

```
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JspCalendar (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JspCalendar (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>
      <% //显示用户名
      if(session.getAttribute("username")!=null){
        out.println(session.getAttribute("username"));
      }
      %> 当前位置: <a href="..index.jsp">首页</a> -&gt; 成员服务 </td>
      <td align="right"><!--显示时间--> <%@ include file="date.inc"%> </td>
    </tr>
    .....HTML 代码省略 (详细代码请参看附带光盘)
  </td>
</tr>
</table>
<!--包含的尾文件-->
<%@ include file="footer.inc"%>
```

## 2. 登录名称及登录源文件

如果用户输入的登录名称已经存在,那么系统会提示用户该用户名称已经存在(如图 15-2 所示)。并且页面将会导向 reg\_again.jsp 页面。

文件 reg\_again.jsp 的源代码如例程 15-2 所示。

例程 15-2

```
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JspCalendar (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JspCalendar (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>
      <% //显示用户名
      if(session.getAttribute("username")!=null){
```

```

out.println(session.getAttribute("username"));
}
%> 当前位置: <a href=" ../index.jsp">首页</a> -&gt; 成员服务 </td>
<td align="right"><!--显示时间--> <%@ include file="date.inc"%> </td>
</tr>
.....HTML 代码省略 (详细代码请参看附带光盘)
<%
    String reg=request.getParameter("reg");
    String errorMessage = "";
    if(reg.equals("error"))
    {
        //提示错误信息
        errorMessage="对不起,这个用户名已经存在!";
    }
%>
<tr><td align="right" height="32" width="40%">
    <font color=red><%=errorMessage%></font>
</td>
</tr>
.....HTML 代码省略 (详细代码请参看附带光盘)

</td>
</tr>
</table>
<!--包含的尾文件-->
<%@ include file="footer.inc"%>

```

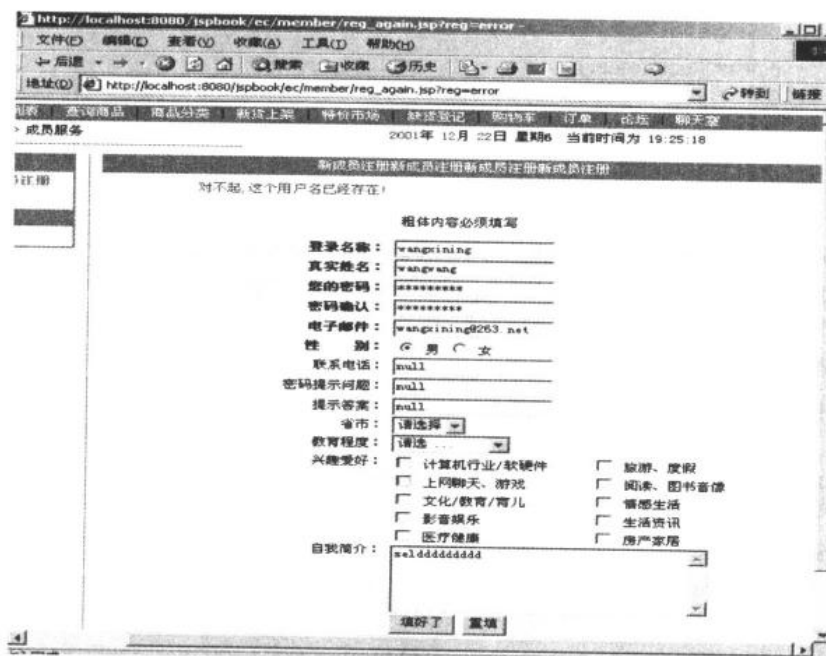


图 15-2 注册失败页面

### 3. 注册成功页面及其源代码

如果输入的用户名没有被其他用户注册使用，并且系统要求填写的信息必须都合理，那么该用户就会成功注册，出现如图 15-3 所示的页面。

文件 reg\_ok.jsp 的源代码如例程 15-3 所示。

例程 15-3

```
<%@ page contentType="text/html;charset=gb2312" %>
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JavaBean （dates.JspCalendar）
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean （test.faq）
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<%!String logname,realname,passwd1,passwd2,E-mail,gender,phone;
String problem,answer,province,education,selfintro,hobby;
String[] hobbies;
boolean regAttempt = false;
String errorMessage = "";
%>
<% //out.print(request.getParameterValues("logname"));
//必须填写的项目
logname=request.getParameter("logname");
realname=request.getParameter("realname");
passwd1=request.getParameter("passwd1");
passwd2=request.getParameter("passwd2");
E-mail=request.getParameter("E-mail");
gender=request.getParameter("Gender");
```

```
//非必须填写的项目
phone=request.getParameter("phone");
if(phone.trim().equals("")){
    phone=null;
}
problem=request.getParameter("problem");
if(problem.trim().equals("")){
    problem=null;
}
answer=request.getParameter("answer");
if(answer.trim().equals("")){
    answer=null;
}
province=request.getParameter("Province");
if(province.trim().equals("")){
    province=null;
}
education=request.getParameter("education");
if(education.trim().equals("")){
    education=null;
}

hobbies=request.getParameterValues("hobbies");
hobby="";
if(hobbies!=null){
    for (int i=0;i<hobbies.length;i++){
        hobby=hobby+hobbies[i];
    }
}else hobby="null";
selfintro=request.getParameter("selfintro");
if(selfintro.trim().equals("")){
    selfintro=null;
}

///转换中文
logname=getStr(logname);
realname=getStr(realname);
passwd1 =getStr(passwd1);
E-mail=getStr(E-mail);
gender =getStr(gender);
phone =getStr(phone);
problem =getStr(problem);
answer =getStr(answer);
province =getStr(province);
education=getStr(education);
hobby=getStr(hobby);
```



```

        selfintro=getStr(selfintro);
    %>
    <% //进行数据库操作
        String sql="select ID from member where logname='"+logname+"'";
        ResultSet RS=workM.executeQuery(sql);
        out.println(sql);
        int rowcount=0;
        try
        {
            while(RS.next())
            {
                rowcount++;
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        if(rowcount==0)
        {    //成功注册
            regAttempt=true;
        }else {
//输入的用户名已经存在
response.sendRedirect("reg_again.jsp?reg=error");
        }
        if(regAttempt==true)
        {
            String sqlinsert;
            sqlinsert="insert into  member(logname,realname,password,E-mail,gender,";
            sqlinsert= sqlinsert+"phone,problem,answer,province,education,hobbies,selfintro) ";
            sqlinsert= sqlinsert+"Values('"+logname+"','"+realname+"','"+passwd1+"','"+E-mail;
            sqlinsert= sqlinsert+"','"+gender+"','"+phone+"','"+problem+"','"+answer;
            sqlinsert= sqlinsert+"','"+province+"','"+education+"','"+hobby+"','"+selfintro+"')";
            //进行数据库操作
            workM.executeQuery(sqlinsert);
            //注册成功
            response.sendRedirect("reg_success.jsp");
        }
    %>
    如果用户成功注册，那么将会导向页面 reg_success.jsp。如图 15.3 所示。
    文件 reg_success.jsp 的源代码如下：
    //头文件 header.inc
    <%@ include file="header.inc"%>
    //调用用来显示时间的 JavaBean （dates.JspCalendar）
    <jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
    <%@ page language="java" import="java.sql.*" %>

```

```
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>
      <% //显示用户名
      if(session.getAttribute("username")!=null){
      out.println(session.getAttribute("username"));
      }
      %> 当前位置: <a href=" ../index.jsp">首页</a> -&gt; 成员服务  </td>
      <td align="right"><!--显示时间--> <%@ include file="date.inc"%> </td> </tr>
      <TR bgColor=#3399ff>
        .....HTML 代码省略 (详细代码请参看附带光盘)

      </td>
    </tr>
  </table>
  <!--包含的尾文件-->
  <%@ include file="footer.inc"%>
```

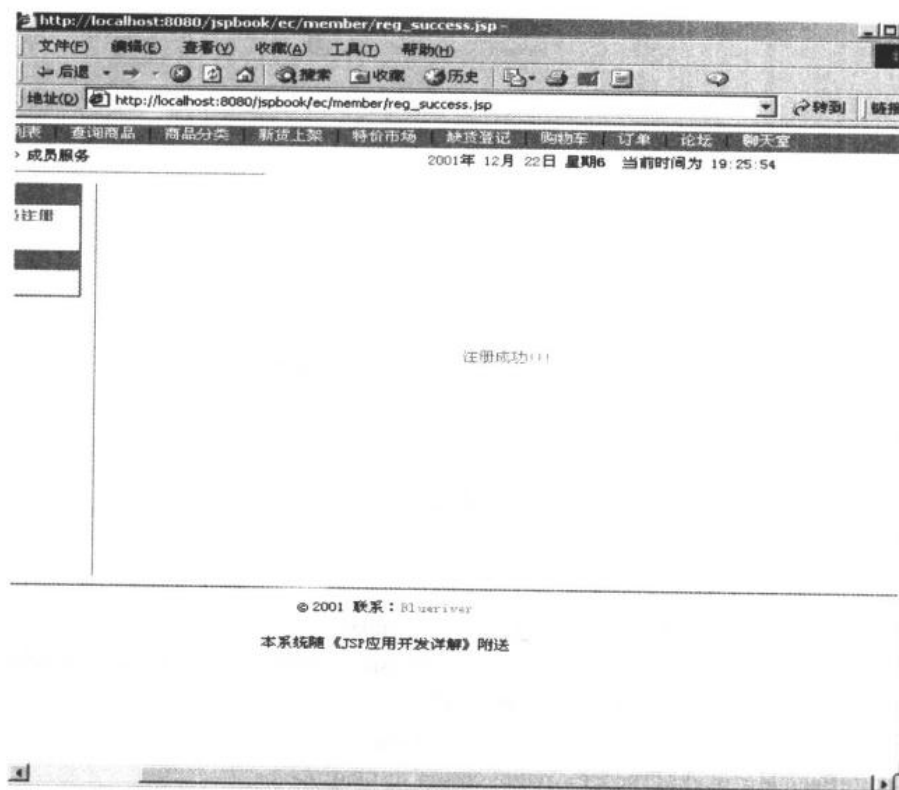


图 15-3 注册成功页面

## 15.2 用户登录系统

该系统采用的登录方式是这样的：

首先，用户在 login.jsp 页面中输入用户名以及密码，输入完毕后，单击【登录】按钮提交登录信息。

提交后，在 login.jsp 页面中进行验证。如果输入的用户名不存在或者输入的用户名与密码不匹配，那么系统会显示错误的提示信息。

如果输入的用户名与密码相匹配，那么系统则会提示用户登录成功。

### 1. 用户登录页面效果图及其源代码

用户登录页面如图 15-4 所示，文件 login.jsp 的源代码如例程 15-4 所示。

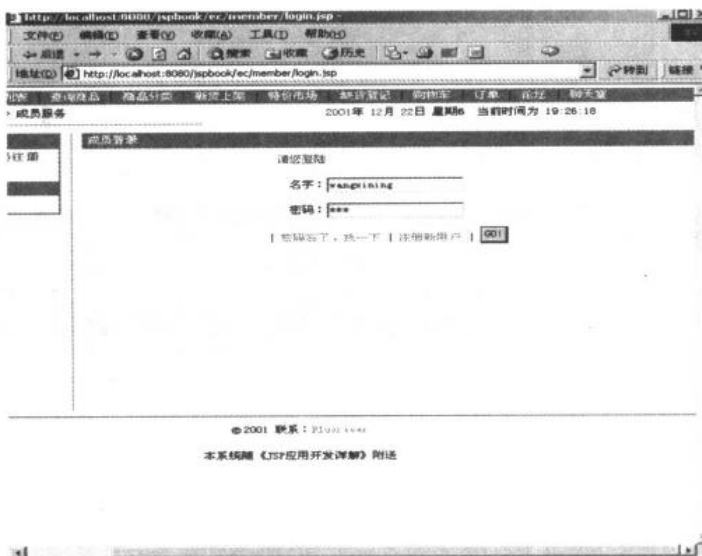


图 15.4 登录界面

例程 15-4

```
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
```

```
try
{
    String temp_p=str;
    byte[] temp_t=temp_p.getBytes("ISO8859-1");
    String temp=new String(temp_t);
    return temp;
}
catch(Exception e)
{
    e.printStackTrace();
}
return "null";
}
%>
<%!
//定义登录时用到的变量并对变量进行初始化
String logname,logpass;
boolean loginAttempt = false;
boolean loginOK = false;
String errorMessage = "请您登录";
%>
<% //判断是否是提交了登录信息
if(request.getParameterValues("login") != null
&& request.getParameterValues("login")[0].trim().equals("GO!"))
&&request.getParameterValues("logname") != null
&&request.getParameterValues("logpass") != null)
{
    loginAttempt = true;
}
if (loginAttempt)
{
    logname=request.getParameter("logname");
    logpass=request.getParameter("logpass");
    logname=getStr(logname);
    logpass=getStr(logpass);
    String sql="select * from member where logname='"+logname+"' and password='"+logpass+"'";
    //执行数据库的操作，查询用户是否存在
    ResultSet RS=workM.executeQuery(sql);
    int rowscount=0;
    try
    {
        while(RS.next())
        {
            rowscount++;
        }
    }
}
```

```

catch(Exception e)
{
    e.printStackTrace();
}
if(rowcount!=0)
{
    //登录成功
    errorMessage="成功登录";
    //设置 session (username) 为输入的用户名
    session.setAttribute("username",logname);
    loginOK=true;
    //判断是否存在参数 url;
    //如果存在, 则页面会导向 url;
    //如果 url 不存在, 那么页面会导向 http://localhost:9090/test/ec
    if(loginOK){
        String url;
        url=request.getParameter("url");
        out.println(url);
        if(url==null){
            response.sendRedirect("http://localhost:9090/test/ec");
        }
        else{
            response.sendRedirect("http://localhost:9090"+url);
        }
    }
}
else{
    //提示登录的错误信息
    errorMessage="您的用户名或者密码不正确";
    //设置 session (username) 为空值
    session.setAttribute("username","");
}
}
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
            if(session.getAttribute("username")!=null){
                out.println(session.getAttribute("username"));
            }
            %> 当前位置: <a href="../index.jsp">首页</a> -&gt; 成员服务 </td>
            <td align="right"><!-- 显示时间--> <%@ include file="date.inc"%> </td>
        </tr>
        <TR bgColor=#3399ff>
            .....HTML 代码省略 (详细代码请参看附带光盘)

            </td>

```

```

</tr>
</table>
<!-- 包含的尾文件 -->
<%@ include file="footer.inc"%>

```

如果输入的用户名和密码不匹配，则会出现提示信息，如图 15-5 所示。

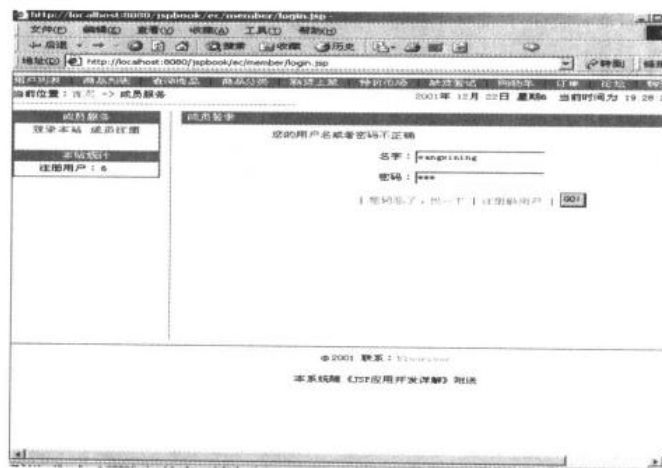


图 15-5 登录的用户名和密码不匹配

如果输入的用户名和密码相匹配，则会成功登录，如图 15-6 所示

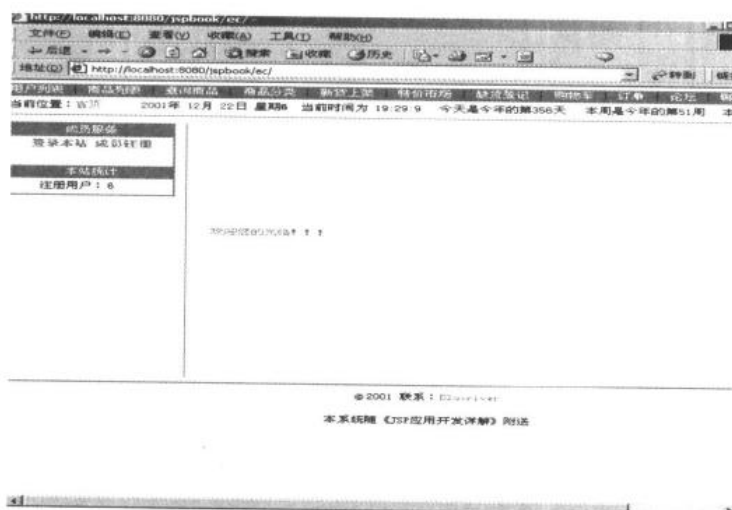


图 15-6 成功登录

## 2. 查询密码效果图及其源代码

如果用户把自己的密码忘记怎么办。没关系，我们的这个系统提供了查询自己密码的功能。在这个系统中，要求用户输入自己的用户名以及注册时输入的 E-mail 地址，提交这些信息后，系统会进行验证。

如果输入的用户名以及 E-mail 地址与注册时的输入相一致，那么系统会在页面上显



示用户的密码。

否则，用户输入的用户名以及 E-mail 地址，与注册时的输入不相一致，那么系统会在页面上显示相应的错误信息。

当然，这种查询密码的方式还不够严密。在实际的开发中，建议系统通过发送 E-mail 的方式，把秘密 E-mail 给用户的信箱。这样的查询密码方式将显得要严密得多，查询密码的效果图如图 15-7 所示。

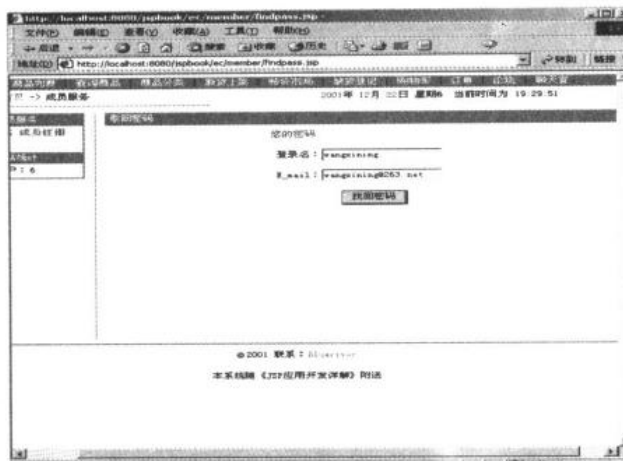


图 15-7 查询密码

文件 findpass.jsp 的源代码如例程 15-5 所示。

例程 15-5

```
<!--包含的头文件-->
<%@ include file="header.inc"%>
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

```

        return "null";
    }
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
<TBODY>
<tr><td align="left" height=25>
<% //显示 session (username) 的值
if(session.getAttribute("username")!=null){
out.println(session.getAttribute("username"));
}
%> 当前位置: <a href="../index.jsp">首页</a> -&gt; 成员服务 </td>
<td align="right"> <%@ include file="date.inc"%> </td>
</tr>
<TR bgColor=#3399ff>
.....HTML 代码省略 (详细代码请参看附带光盘)

<td width="100%" colspan="2" height="20" bgcolor="#3399ff">&nbsp;  <font
color="#ffffff">取回密码</font>
</td>
</tr>
<%! //定义变量并进行初始化
String logname,E-mail;
boolean loginAttempt = false;
String errorMessage = "";
%>
<% //判断是否提交信息
if(request.getParameterValues("findpass") != null
&& request.getParameterValues("findpass")[0].trim().equals("找回密码")
&& request.getParameterValues("logname") != null
&& request.getParameterValues("E-mail") != null)
{
    loginAttempt = true;
}
if (loginAttempt)
{
    //进行数据库操作, 查询密码
    logname=request.getParameter("logname");
    E-mail=request.getParameter("E-mail");
    logname=getStr(logname);
    E-mail=getStr(E-mail);
    String sql="select * from member where logname='"+logname+"' and E-mail='"+E-mail+"'";
    ResultSet RS=workM.executeQuery(sql);
    int rowscount=0;
    try
    {
        while(RS.next())
        {

```

```
        rowcount++;
        errorMessage=RS.getString("password");
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
//如果输入的用户名与密码不匹配，则会提示错误信息
if(rowcount!=0)
{
    errorMessage=RS.getString("password");
}else errorMessage="您的用户名或者 E-mail 不正确";
}
%>

|  |
| --- |
| <font color=red>您的密码:<%=errorMessage%></font> </td> .....HTML 代码省略（详细代码请参看附带光盘） </td> </tr> </table> <!--包含的尾文件--> <%@ include file="footer.inc"%> |

```

如果输入的登录名和 E-mail 相匹配，则会显示您的密码，如图 15-8 所示。

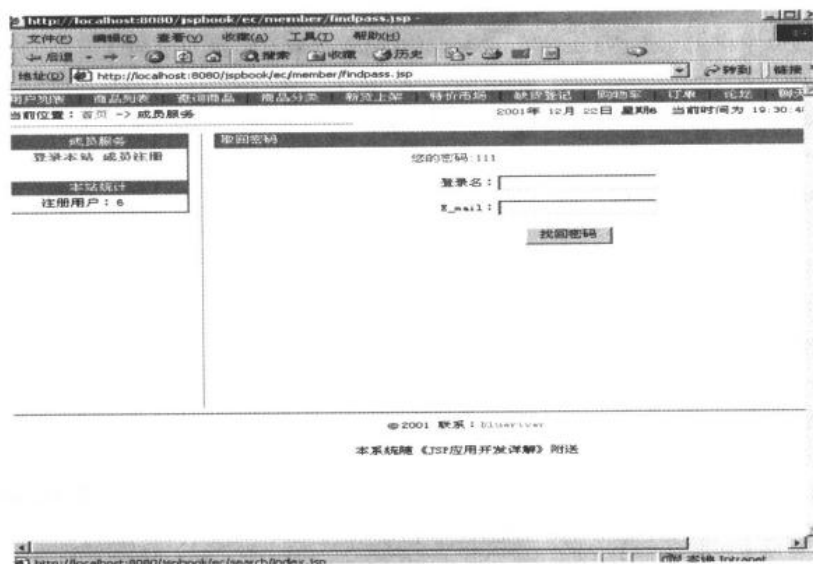


图 15-8 显示您的密码

如果输入的登录名和 E-mail 不匹配，则会提示您的用户名或者 E-mail 不正确，如图 15-9 所示。

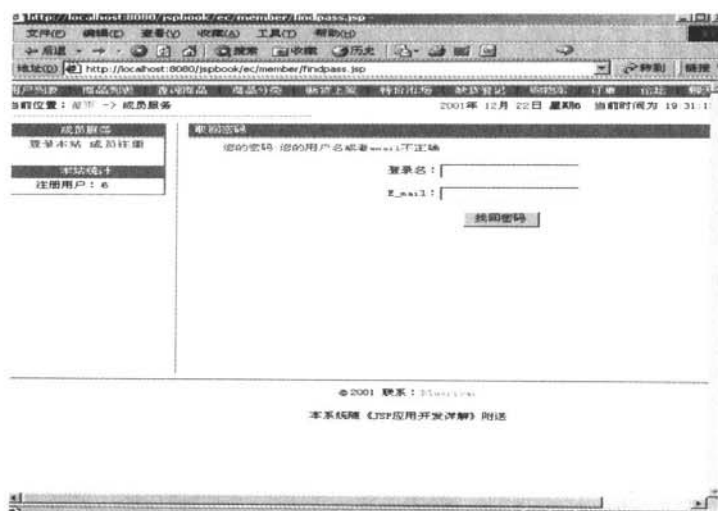


图 15-9 提示您的用户名或者 E-mail 不正确



# 第 16 章 查 询 商 品

## 1. 查询商品页面效果图及其源代码

商品的查询系统是这样实现的：

首先，用户在页面 search/index.jsp 中输入要查询的商品名称的关键字，并且选择查询的范围。然后，单击【查询】按钮进行查询。

查询的事务逻辑将在 search/search.jsp 页面中进行，查询商品的页面如图 16-1 所示。

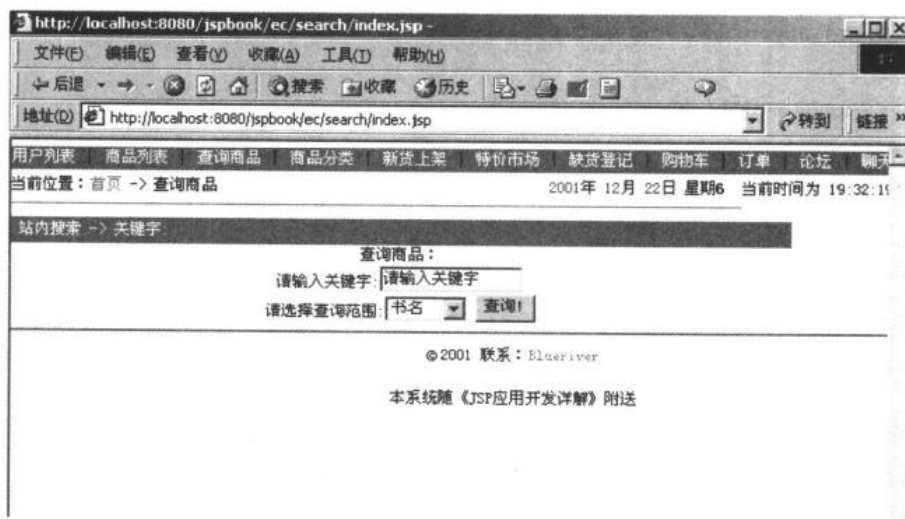


图 16-1 查询商品

search/index.jsp 源代码如例程 16-1 所示。

例程 16-1

```
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
```



```

        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>当前位置: <a href="../index.jsp">首页</a> -&gt; 查询商品
    </td>
        <%@ include file="../member/date.inc"%>
    </tr>
        <TR bgColor=#3399ff>
            .....HTML 代码省略（详细代码请参看附带光盘）
        </td>
    </tr>
</table>
<%@ include file="../member/footer.inc"%>

```

## 2. 页面逻辑处理代码

查询商品的事务逻辑将在 search/search.jsp 页面中进行。  
search/search.jsp 页的源代码如例程 16-2 所示。

例程 16-2

```

//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JavaBean （dates.JspCalendar）
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type='dates.JspCalendar' />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean （test.faq）
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
}

```

```

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<%!
public static String category(int id)
{
    try
    {
        switch(id)
        {
            case 1:
                return "计算机类";
            case 2:
                return "英语类";
            default:
                return "其他类";
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>当前位置: <a href="../index.jsp">首页</a> -&gt; 查询商品
</td>
    <%@ include file="../member/date.inc"%>
  </tr>
  <TR bgColor=#3399ff>
    <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
    <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
  </TBODY></TABLE>
<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
  <tr>

```

398

```

if(way.trim().equals("3")){
    strSQL="SELECT * FROM book where";
    strSQL=strSQL+" publish_name like '%" +keyword+"%'";
}

ResultSet RSa = workM.executeQuery(strSQL);
int searchnum;
searchnum=0;
while (RSa.next()) {
    String srch_title,author;
    int id,cate;
id=RSa.getInt("id");
    srch_title=getStr(RSa.getString("name"));
    author=getStr(RSa.getString("author"));
    cate=RSa.getInt("category");
    out.println("<tr height='23'><td>");
    out.println("<li><a href=javascript:show('../book_store/book.jsp?id=");
out.println(id+"','详细资料',480,500)>"+srch_title+"</a>");
    out.println("</td>");
    out.println("<td>"+author+"&nbsp;</td>");
    out.println("<td>"+category(cate)+"</td>");
    out.println("<td><a href=javascript:show('../book_store/book.jsp?id";
out.println("="+id+"','详细资料',480,500)>[详细资料]</a></td>");
    out.println("</tr>");

    searchnum=searchnum+1;
}
RSa.close();
out.println("</ul>");
out.println("共搜索到<b><font color=Red>"+searchnum+"</font></b>条纪录");
%>
<p align="right"><a href="javascript:history.go(-1)">返回</a></p>
</td>
<td width="5%"></td>
</tr>
</table>
</td>
</tr>
</table>
<%@ include file="../member/footer.inc"%>

```

如果存在与输入的关键字相匹配的记录，则会出现如图 16-2 所示的界面。

相反，如果不存在与输入的关键字相匹配的记录，则会出现如图 16-3 所示的界面。

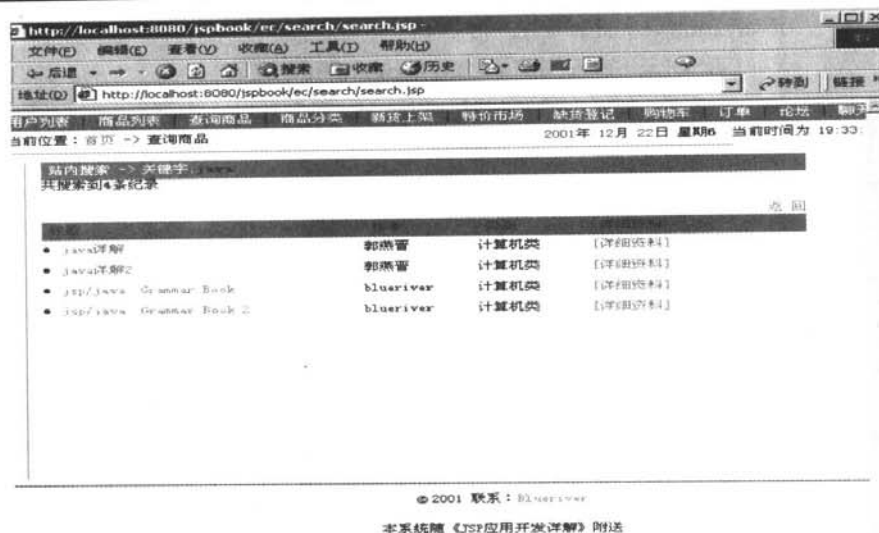


图 16-2 找到符合条件的记录

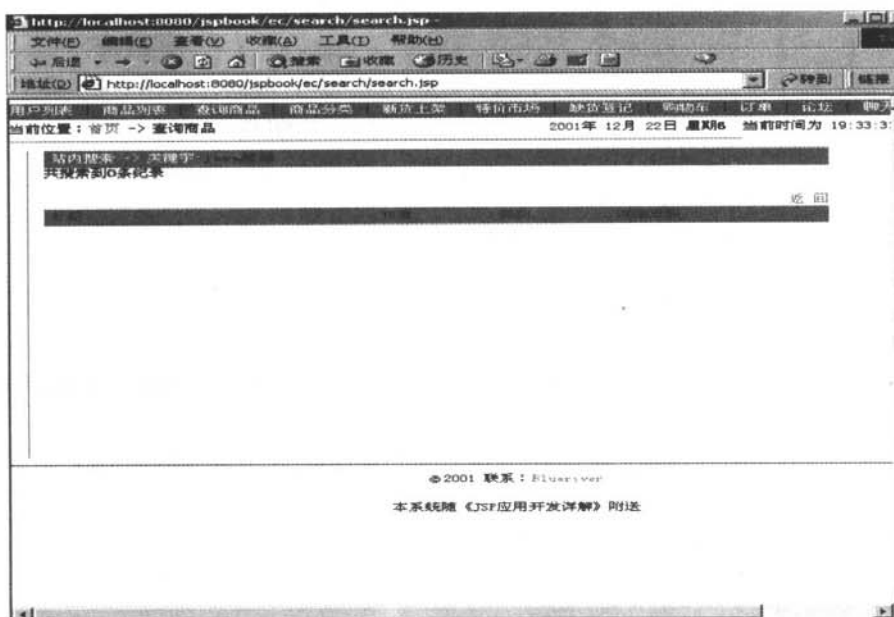


图 16-3 没有找到符合条件的记录

# 第 17 章 商品分类

## 1. 商品分类页面效果图及其源代码

在本系统中，商品（书）可以按照类别进行分类。单击导航栏上的商品分类链接，将会出现如图 17-1 所示的界面。在该页面中，显示了各种类别的商品（书）并且显示了各种商品的数量。

在实际的项目开发中，商品的种类将会异常的复杂。比如说，在 8848 网站的商品列表中，就列举了大量的种类。在本例子中，我们只是通过讲解简单的实例，使读者能够在实际的应用中举一反三。商品分类界面如图 17-1 所示。

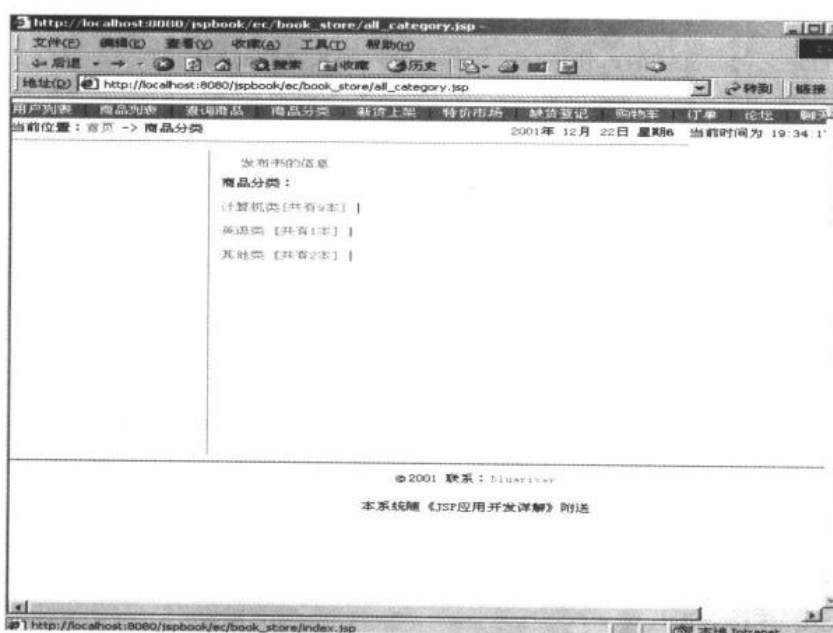


图 17-1 商品分类

文件 all\_category.jsp 的源代码如例程 17-1 所示。

### 例程 17-1

```
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JavaBean （dates.JspCalendar）
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean （test.faq）
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
```



```

public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<%! //自定义的静态函数，作用是：用数字分别来表示商品的种类。
public static String category(int id)
{
    try
    {
        switch(id)
        {
            case 1:
                return "计算机类";
            case 2:
                return "英语类";
            default:
                return "其他类";
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
            if(session.getAttribute("username")!=null){
            out.println(session.getAttribute("username"));
            }
            %> 当前位置: <a href="../index.jsp">首页</a> -&gt; 商品分类    </td>

```

```

<!--显示时间--><%@ include file="../member/date.inc"%> </tr>
<TR bgColor=#3399ff>
    <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></TD></TR>
<tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
width=16></td></tr>
</TBODY></TABLE>
<table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
    <tr>
        <td width="150" height="355" valign="top"></td>
        <td width="10" height="100%"></td>
        <td width="1" height="100%" bgcolor="#3399ff"></td>
        <td width="10" height="100%"></td>
        <td width="589" height="331" valign="top" background="images/bg1.gif">
<table border="0" width="100%" cellspacing="0" cellpadding="0">
            <tr><td height="1" bgcolor="" colspan="2">
                <td align="left" colspan="2" height="32">
                    <a href="post.jsp">发布书的信息</a>
                </td>
            </tr>
        </table>
    </td>
</tr>
<%
//sql1、sql2、sql3 分别是用来查询计算机类、英语类、其他类书的数量
String sql1="select * from book where category=1";
String sql2="select * from book where category=2";
String sql3="select * from book where category=3";
//n1、n2、n3 分别是用来纪录计算机类、英语类、其他类书的数量
int n1;
int n2;
int n3;
//初始化 n1、n2、n3
n1=0;
n2=0;
n3=0;
//查询计算机类的书籍
ResultSet RSize1 = workM.executeQuery(sql1);
while(RSize1.next()){
    n1=n1+1;
}
//查询英语类的书籍
ResultSet RSize2 = workM.executeQuery(sql2);
while(RSize2.next()){
    n2=n2+1;
}
//查询其他类的书籍
ResultSet RSize3 = workM.executeQuery(sql3);

```

```
while(RSsize3.next()){
    n3=n3+1;
}
%>
    商品分类: <br><br>
<a href="category.jsp?id=1" target="_blank">计算机类[共有<%=n1%>本]</a> |<br><br>
<a href="category.jsp?id=2" target="_blank">英语类 [共有<%=n2%>本]</a> |<br><br>
<a href="category.jsp?id=3" target="_blank">其他类 [共有<%=n3%>本]</a> |<br><br>
</td> </tr> </table>
<!--包含的尾文件-->
<%= include file="footer.inc"%>
```

## 2. 查看每种类别的列表

单击如图 17-1 所示中的每种商品种类的链接, 将会出现该种类商品列表, 如图 17-2 所示。在图 17-2 中, 显示了计算机类书籍的列表, 读者不难看出该种商品列表有多个页面。

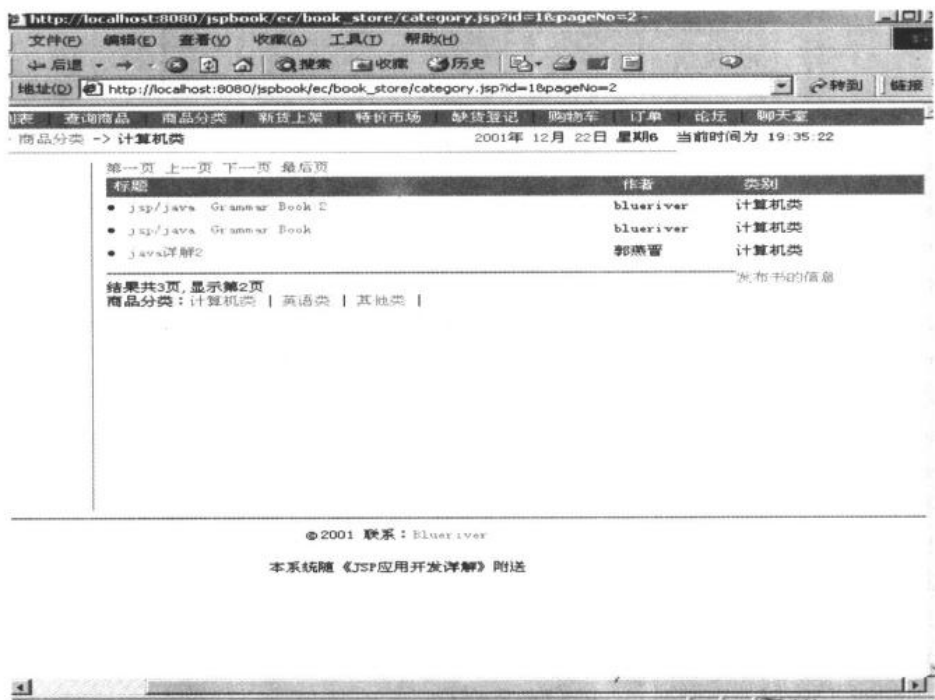


图 17-2 查看每种类别的商品

读者可以重点学习这种分页方式, 在本系统中, 有多处使用了这种分页方式。文件 category.jsp 源代码如例程 17-2 所示。

### 例程 17-2

```
//头文件 header.inc
<%= include file="header.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
```

```
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>

<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>
      <% //显示用户名
      if(session.getAttribute("username")!=null){
      out.println(session.getAttribute("username"));
      }
      %> 当前位置: <a href="../index.jsp">首页</a> -&gt; <a href="all_category.jsp">商品分类</a>
      <%
      //通过参数传递的 id 是 String 型
      String dirname;
      Integer dir=new Integer(request.getParameter("id"));
      //由于 switch 中只能接受整数型, 因此需要将传递的参数 id 转化为整数型
      switch(dir.intValue())
      {
          case 1:
              dirname="计算机类 ";
              break;
          case 2:
              dirname=" 英语类 ";
              break;
          default:
              dirname=" 其他类 ";
              break;
      }
      %>
```



```

        if((t%3)>0){
            mtotal=t/3+1;
        }else mtotal=t/3;
    %>
    <%!String pageNo, mTmp;
        int i, j, k;
    %>
    <% //pageNo 表示请求的是第几页
        pageNo = request.getParameter("pageNo");
        //如果 pageNo 为 null 值, 则赋值为 1
        if(pageNo == null){
            pageNo = "1";
        }
        //j 表示的是 pageNo 对应的整型值
        j = Integer.parseInt(pageNo);
        //当 j 的值小于 1 时, 赋值为 1
        if(j < 1)
            j = 1;
        //当 j 的值大于总页数时, 赋值为 mtotat
        if(j > mtotat)
            j = mtotat;
    %>
    <% // 读取数据库记录
        String strSQL="SELECT * FROM book where category="+cate+" order by id desc";
        ResultSet RSa = workM.executeQuery(strSQL);
        //记录集移动到相应的位置
        //j 为逻辑页数
        for(k = 0;k < (j-1)*3;k++)
        {
            RSa.next();
        }
        i = 0;
        k = 1;
        while (RSa.next()) {
            i = i + 1;
            //超过 3 条
            if(i == 4)
            {
                k = 0;
                break;
            }
            out.print("<tr height='23'><td><li><a href=book.jsp?id="+RSa.getInt("id");
            out.print(">"+getStr(RSa.getString("name"))+"</a></td><td>");
            out.print(getStr(RSa.getString("author"))+"</td><td>");
            out.print(category(RSa.getInt("category"))+"</td><td></td></tr>");
        }
    
```



```

        i = i - k;
//关闭记录集
        RSa.close();
    %>
    <% //以下代码行是用来显示页面数的
//当 j 大于 1 时, 就会显示 “第一页”、“上一页” 字样
        if(j > 1)
        {
    %>
            <a href="category.jsp?id=<%=cate%>&pageNo=1">第一页</a>
    <%
            int ii = Integer.parseInt(pageNo,10);
            if(ii > 1)
                ii = ii - 1;
            String ssTmp = Integer.toString(ii);
    %>
            <a href="category.jsp?id=<%=cate%>&pageNo=<%=ssTmp%>">上一页</a>
    <%
        }
//同样地, 当 j 小于总逻辑页数时, 就会显示 “下一页”、“最后页” 字样
        if(j < mttotal)
        {
            int ii = Integer.parseInt(pageNo,10);
            if(ii < mttotal)
                ii = ii + 1;
            String ssTmp = Integer.toString(ii);
    %>
            <a href="category.jsp?id=<%=cate%>&pageNo=<%=ssTmp%>">下一页</a>
            <a href="category.jsp?id=<%=cate%>&pageNo=<%=mttotal%>">最后页</a>
    <%
        }
        if(mttotal < j)
            j = mttotal;
    %>

        <tr><td height="1" bgcolor="" colspan="2">
        <hr color="#3399ff">
        结果共<%=mttotal%>页,显示第<%=j%>页</td>
        <td align="left" colspan="2" height="32">

```

以下代码省略, 详细代码请参考附带光盘的源代码.....

### 3. 查看商品的详细资料

单击每种商品名称, 则会出现如图 17-3 所示的界面。在图 17-3 所示的界面中, 显示了相应的书籍的详细信息。

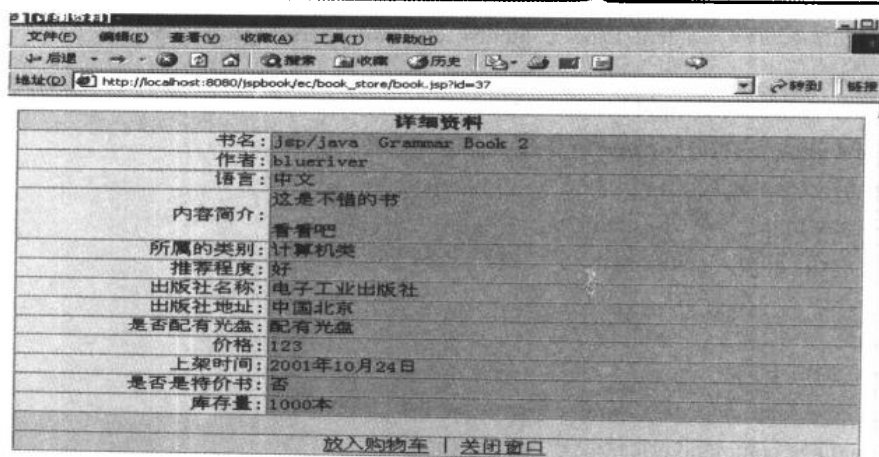


图 17-3 商品的详细资料

文件 book.jsp 源代码如例程 17-3 所示。

例程 17-3

```
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<html>
<head>
<title>[详细资料]</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

```

<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body bgcolor="#FFFFFF">
<table width="100%" border="1" cellspacing="0">
  <tr bgcolor="#CCCCFF">
    <td width="21%" colspan=2>
      <div align="center"><b>详细资料</b></div>
    </td>
  </tr>
  <%! //声明变量
String name,author,content,publish_name,publish_address;
int language,commend,price,book_number,category;
String on_sale_time;
boolean good_price,cdrom;
%>
    <%      //执行数据库操作，显示书籍的详细信息
      String id;
      id=request.getParameter("id");
      String strSQL="SELECT * FROM book where id="+id;
      ResultSet RSa = workM.executeQuery(strSQL);
      while (RSa.next()) {
        name=getStr(RSa.getString("name"));
        author=getStr(RSa.getString("author"));
        language=(RSa.getInt("language"));
        content=getStr(RSa.getString("content"));
        category=(RSa.getInt("category"));
        commend=(RSa.getInt("commend"));
        publish_name=getStr(RSa.getString("publish_name"));
        publish_address=getStr(RSa.getString("publish_address"));
        cdrom=(RSa.getBoolean("cdrom"));
        price=(RSa.getInt("price"));
        on_sale_time=getStr(RSa.getString("on_sale_time"));
        good_price=(RSa.getBoolean("good_price"));
        book_number=(RSa.getInt("book_number"));
      }
      RSa.close();
    %>
  <tr>
    <td bgcolor="#FFCCCC" width="21%">
      <div align="right">书名:</div>
    </td>
    <td bgcolor="#33FFCC" width="79%"><%=name%></td>
  </tr>
  <tr>
    <td bgcolor="#FFCCCC" width="21%">
      <div align="right">作者:</div>

```

```
</td>
<td bgcolor="#33FFCC" width="79%"><%=author%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
<div align="right">语言:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=lang(language)%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="30%">
<div align="right">内容简介:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=content%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
<div align="right">所属的类别:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=category(category)%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
<div align="right">推荐程度:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=commend(commend)%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
<div align="right">出版社名称:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=publish_name%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
<div align="right">出版社地址:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=publish_address%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
<div align="right">是否配有光盘:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=cd(cdrom)%></td>
</tr>
<tr>
```

```
<td bgcolor="#FFCCCC" width="21%">
    <div align="right">价格:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=price%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
    <div align="right">上架时间:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=YMD(on_sale_time)%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
    <div align="right">是否是特价书:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=price(good_price)%></td>
</tr>
<tr>
<td bgcolor="#FFCCCC" width="21%">
    <div align="right">库存量:</div>
</td>
<td bgcolor="#33FFCC" width="79%"><%=book_number%>本</td>
</tr>
<tr bgcolor="#CCCCFF">
    <td colspan=2>&nbsp;   </td>
</tr>
<tr bgcolor="#CCCCFF">
    <td colspan=2 align="center"><a href="shopcart.jsp?book_id=<%=id%>">放入购物车</a> | <a
href="javascript:close();">关闭窗口</a></td>
</tr>
</table>
</body>
</html>
```

#### 4. 查看总的商品列表

文件 `book_store/index.jsp` 的源代码与 `category.jsp` 的源代码基本类似, 惟一的不同是执行数据库操作的 SQL 语句不同, 这也是两者显示不同内容的原因。而除此之外, 其他代码基本相同。

文件 `book_store/index.jsp` 的源代码可以参看附带光盘。在这里不再列出。其执行的显示效果如图 17-4 所示。

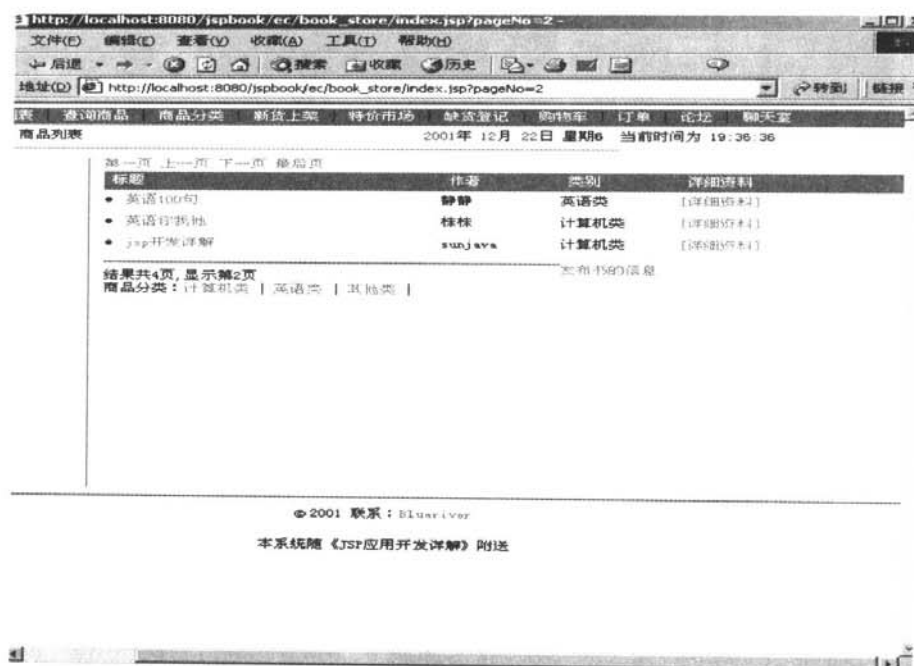


图 17-4 总的商品列表





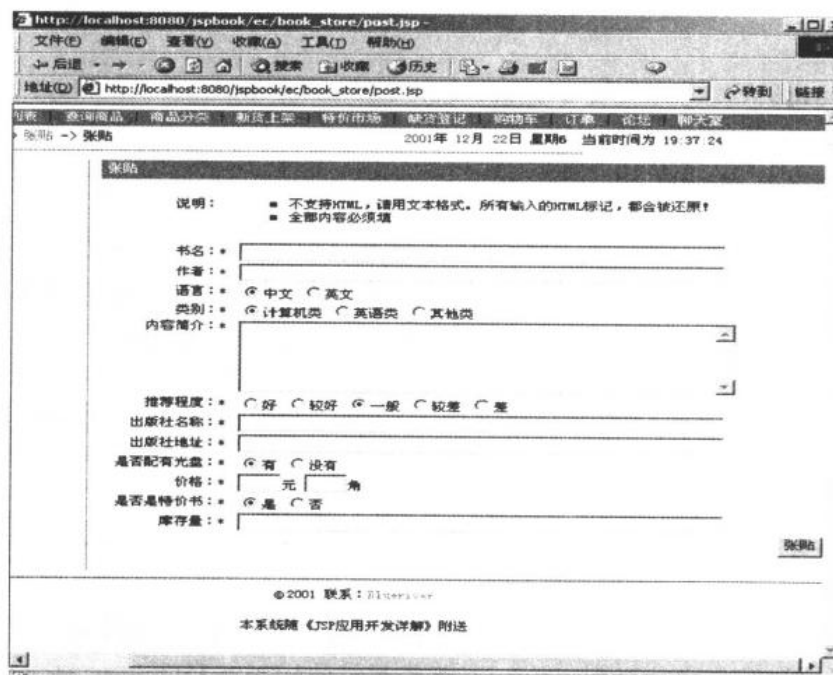
# 第 18 章 最新、特价及缺货商品列表

## 18.1 查看最新发布商品列表

### 1. 发布新商品的页面效果图及其源代码

在实际的项目开发中，类似发布新商品等属于后台管理员操作的事务，应该在单独的管理页面中进行。在本例子中，我们把这个发布新商品的功能在代码前端显示了，在这里的目的只是让读者明白管理事务的一般方法。在实际的应用开发中，建议读者应该根据实际情况，做相应的变动。

在如图 18-1 所示的界面中，显示了发布新商品时必须输入的商品的资料。



The screenshot shows a web browser window with the address bar displaying 'http://localhost:8080/jspbook/ec/book\_store/post.jsp'. The page title is '发布新商品' (Post New Product). The form contains the following fields and options:

- 说明:
  - ☒ 不支持HTML，请用文本格式。所有输入的HTML标记，都会被还原！
  - ☒ 全部内容必须填
- 书名: \*
- 作者: \*
- 语言: \* ☒ 中文 ☐ 英文
- 类别: \* ☒ 计算机类 ☐ 英语类 ☐ 其他类
- 内容简介: \*
- 推荐程度: \* ☐ 好 ☐ 较好 ☒ 一般 ☐ 较差 ☐ 差
- 出版社名称: \*
- 出版社地址: \*
- 是否配有光盘: \* ☒ 有 ☐ 没有
- 价格: \* 元 角
- 是否是特价书: \* ☒ 是 ☐ 否
- 库存量: \*

At the bottom of the form is a '张贴' (Post) button. The footer of the page includes the copyright notice '© 2001 联系: jilic@163.com' and the text '本系统随《JSP应用开发详解》附送'.

图 18-1 发布新商品的页面

文件 `book_store/post.jsp` 的源代码如例程 18-1 所示。

例程 18-1

```
//头文件 header.inc
<%@ include file="head.inc"%>
<% //判断用户是否已经登录；如果还没有登录，则页面会直接导向登录界面；
    //在实际的开发中，还应该判断登录的用户是否具有执行发布新的商品的权限；
    //也就是说，只有具有相应的权限的用户，才可以进行下面的操作；
```

```
if(session.getAttribute("username")==null){
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
            if(session.getAttribute("username")!=null){
                out.println(session.getAttribute("username"));
            }
            %>当前位置: <a href="../index.jsp">首页</a>-&gt; <a href="index.jsp">张贴</a> -&gt; 张贴</td>
            <%@ include file="../member/date.inc"%>
        </tr>
        <TR bgColor=#3399ff>
            .....HTML 代码省略(详细代码请参看附带光盘)
        </td>
    </tr>
</table>
<!--包含的尾文件-->
<%@ include file="footer.inc"%>
```



发布新商品的页面应该是在后台管理中进行,在该例子中没有考虑用户是否具有发布新商品的权限。在该例子中,只要是登录用户就可以发布新商品。如果没有登录用户,则会提示登录。代码如下:

```
<%if(session.getAttribute("username")==null){
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI())
;
}
%>
```

## 2. 页面逻辑处理代码

在填写完发布新商品的资料之后,单击【发布】按钮,提交资料到数据库中,然后出现如图 18-2 所示的发布成功页面。执行该业务逻辑的代码在页面 post\_ok.jsp 中。

另外,需要注意的是,如果在信息传递过程中,没有对 HTML 标记符号进行处理,则可能会出现一些令人啼笑皆非的事情,譬如说,您若在输入框中输入如下的代码:

```
<input type=submit value=“我可以发布一个按钮,您信吗??”>
```

然后,单击【发布】按钮,则会在前端的页面上显示一个真的按钮。而这并不是我们需要的,因此必须对输入的 HTML 语言进行处理。

如果不对 HTML 语言进行处理,可能会发生更为糟糕的事情。前段时间,网上流行的一个 JavaScript 语句(其实是调用了一个注册过的 Applet),调用的 Applet 代码如下:

```
<Applet Height=0 Width=0 code=com.ms.activeX.ActiveXComponent></APPLET>
```

.....其余代码省略

如果不对输入的信息进行处理,则可能会发生这样糟糕的事情:这段代码运行,并且修改了您的注册表,直接的效果就是在您的浏览器标题上可能会显示作弊者的行为结果(读者可能在浏览一些网站时遇到过)。

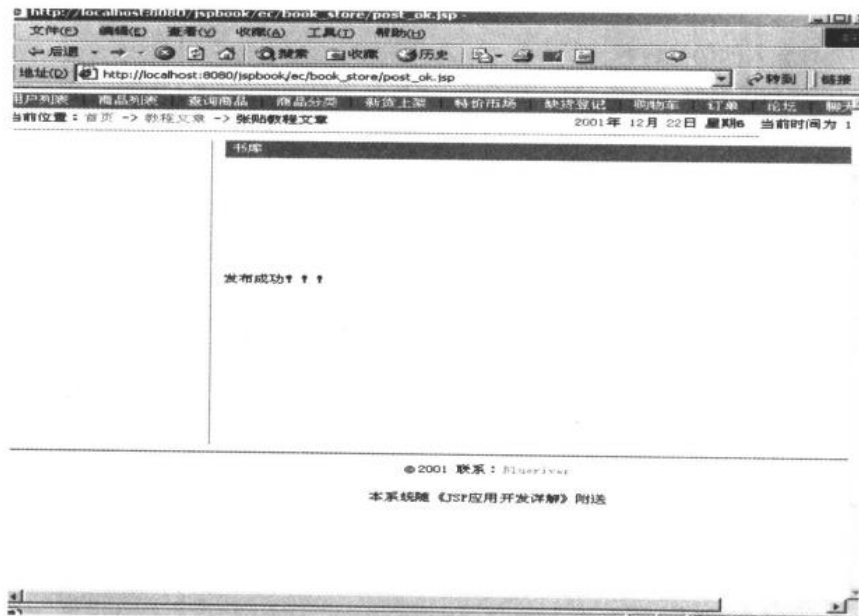


图 18-2 发布成功页面

文件 post\_ok.jsp 的源代码如例程 18-2 所示。

#### 例程 18-2

```
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
    }
}
```

```

    {
        e.printStackTrace();
    }
    return "null";
}
%>
<%! //处理输入的信息内容
    //该函数能够将字符串 sStr 中的'\n'、'\r'转换为<br>;
    //也能够将字符串 sStr 中的' '转换为&nbsp;;
public static String returnToBr(String sStr)
//如果 sStr 为 null 或者 sStr 为空值;
//则该函数会返回不发生变化的 sStr;
if (sStr == null || sStr.equals(""))
return sStr;
}
//定义 StringBuffer 型的变量 sTmp
StringBuffer sTmp = new StringBuffer();
int i = 0;
while (i <= sStr.length()-1)
//将字符串 sStr 中的'\n'、'\r'转换为<br>;
if (sStr.charAt(i) == '\n' || sStr.charAt(i) == '\r')
sTmp = sTmp.append("<br>");
} else if (sStr.charAt(i) == ' ')
将字符串 sStr 中的' '转换为&nbsp;;
sTmp = sTmp.append("&nbsp;");
} else
sTmp = sTmp.append(sStr.substring(i,i+1));
}
i++;
}
String S1;
S1=sTmp.toString();
return S1;
}

%>
<%! //处理输入的信息内容
    //该函数能够将字符串 sStr 中的'<' 转换为"&lt;";
    //也能够将字符串 sStr 中的'>' 转换为"&gt;";
public static String returnToHTML(String sStr)
//如果 sStr 为 null 或者 sStr 为空值;
//则该函数会返回不发生变化的 sStr;
if (sStr == null || sStr.equals(""))
return sStr;
}
//定义 StringBuffer 型的变量 sTmp

```

```

StringBuffer sTmp1 = new StringBuffer();
int i = 0;
while (i <= sStr.length()-1)
//该函数能够将字符串 sStr 中的'<' 转换为"&lt;";
if (sStr.charAt(i) == '<')
sTmp1 = sTmp1.append("&lt;");
//也能够将字符串 sStr 中的'>' 转换为"&gt;";
} else if (sStr.charAt(i) == '>')
sTmp1 = sTmp1.append("&gt;");
}else
{
sTmp1 = sTmp1.append(sStr.substring(i,i+1));
}
i++;
}
String S2;
S2=sTmp1.toString();
return S2;
}
%>
<%! //定义变量，以便用于保存提交上来的商品资料
String name,author,content,category,publish_name,publish_address;
String language,commend,price,book_number;
String cdrom,good_price;
%>
<% //应用函数 returnToHTML 对提交的信息进行处理
name= returnToHTML(request.getParameter("name"));
author= returnToHTML(request.getParameter("author"));
content= returnToHTML(request.getParameter("content"));
category= returnToHTML(request.getParameter("category"));
publish_name= returnToHTML(request.getParameter("publish_name"));
publish_address= returnToHTML(request.getParameter("publish_address"));
language= returnToHTML(request.getParameter("language"));
commend= returnToHTML(request.getParameter("commend"));
price= returnToHTML(request.getParameter("price1")+ "." +request.getParameter("price2"));
book_number= returnToHTML(request.getParameter("book_number"));
cdrom= returnToHTML(request.getParameter("cdrom"));
good_price= returnToHTML(request.getParameter("good_price"));
//应用函数 returnToBr 对提交的信息进行处理
name=returnToBr(name);
author=returnToBr(author);
content=returnToBr(content);
category=returnToBr(category);
publish_name=returnToBr(publish_name);
publish_address=returnToBr(publish_address);
language=returnToBr(language);

```



```

commend=returnToBr(commend);
price=returnToBr(price);
book_number=returnToBr(book_number);
cdrom=returnToBr(cdrom);
good_price=returnToBr(good_price);
//对提交的信息进行中文处理
name=getStr(name);
author=getStr(author);
content=getStr(content);
category=getStr(category);
publish_name=getStr(publish_name);
publish_address=getStr(publish_address);
%>
//头文件 header.inc
<%@ include file="head.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>
      <% //显示用户名
      if(session.getAttribute("username")!=null){
        out.println(session.getAttribute("username"));
      }
      %> 当前位置: <a href=" ../index.jsp">首页</a> -&gt; <a href="index.jsp">教程文章</a> -&gt; 张
      贴教程文章 </td>
    <!--显示时间--><%@ include file=" ../member/date.inc"%> </tr>
    <TR bgColor=#3399ff>
      <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
      width=16></TD></TR>
      <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
      width=16></td></tr>
    </TBODY></TABLE>
    <table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
      <tr>
        <td width="150" height="355" valign="top"> </td>
        <td width="10" height="100%"></td>
        <td width="1" height="100%" bgcolor="#3399ff"></td>
        <td width="10" height="100%"></td>
        <td width="589" height="331" valign="top" background="images/bg1.gif">
          <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
            <tr><td width="100%" height="20" bgcolor="#3399ff">&nbsp;
            <font color="#ffffff">书库</font>
          </td> </tr>
          <tr> <td>
            <%out.println("发布成功!!! ");%>

```

```

</td> </tr> </table></td> </tr> </table>
<%//执行数据库操作，发布新的商品
String sqlinsert="insert into book(name,author,language,content,category";
sqlinsert= sqlinsert+ ",commend,publish_name, publish_address,cdrom,";
sqlinsert= sqlinsert+"price,good_price,book_number) ";
sqlinsert= sqlinsert+ Values("'" +name+"','" +author+"','" +language+"','";
sqlinsert= sqlinsert+content+"','" +category+"','" +commend+"','" +publish_name;
sqlinsert= sqlinsert+"','" +publish_address+"','" +cdrom+"','" +price;
sqlinsert= sqlinsert+"','" +good_price+"','" +book_number+"')";
workM.executeQuery(sqlinsert);
%><!--尾文件-->
<%@ include file="footer.inc"%>

```

### 3. 查看最新发布的 10 种商品

如果浏览者想要查看最近的新商品，那么可以通过单击导航栏上的“新货上架”来查看最新发布的 10 种商品，则出现如图 18-3 所示的页面。

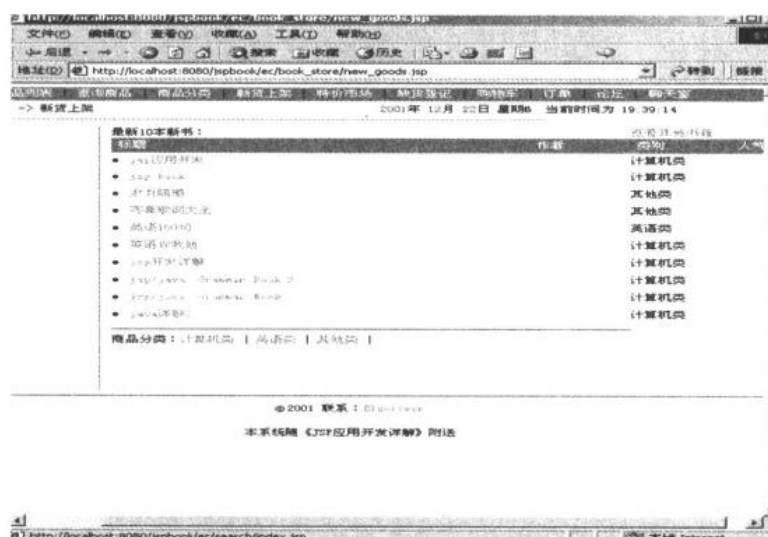


图 18-3 查看最新发布的 10 种商品

文件 new\_goods.jsp 的源代码如例程 18-3 所示。

例程 18-3

```

//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JaveBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JaveBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)

```

```

{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>

```

<%!//自定义的静态函数，作用是：用数字分别来表示商品的种类。

```
public static String category(int id)
```

```

{
    try
    {
        switch(id)
        {
            case 1:
                return "计算机类";
            case 2:
                return "英语类";
            default:
                return "其他类";
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>

```

```
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
```

```
<TBODY>
```

```
<tr><td align="left" height=25>
```

```
<% //显示用户名
```

```
if(session.getAttribute("username")!=null){
```

```
out.println(session.getAttribute("username"));
```

```
}
```

```
%> 当前位置: <a href="../index.jsp">首页</a> -&gt; 商品分类 </td>
```

```
<!--显示时间--><%@ include file="../member/date.inc"%> </tr>
```

```

<TR bgColor=#3399ff>
.....HTML 代码省略（详细代码请参看附带光盘）
<% //执行数据库操作，查询最新的 10 种书籍
String strSQL="SELECT top 10 * FROM book order by on_sale_time desc";
ResultSet RSa = workM.executeQuery(strSQL);
while (RSa.next()) {
    out.print("<tr height='23'><td><li><a href=javascript:show('book.jsp?id=");
out.print(RSa.getInt("id")+";'详细资料',480,500)");
    out.print(">"+getStr(RSa.getString("name"))+"</a></td><td></td><td>");
out.print( (category(RSa.getInt("category")))+"</td><td></td></tr>");
}
RSa.close();
%>
.....HTML 代码省略（详细代码请参看附带光盘）
商品分类: <a href="category.jsp?id=1">计算机类</a> |
<a href="category.jsp?id=2">英语类 </a> |
<a href="category.jsp?id=3">其他类 </a> |
</td>
</tr>
</table>
<!--尾文件-->
<%@ include file="footer.inc"%>

```

## 18.2 查看特价商品列表

如果浏览者想要查看最近特价商品的列表，那么可以通过单击导航栏上的特价市场来查看特价商品的全部列表，如图 18-4 所示。

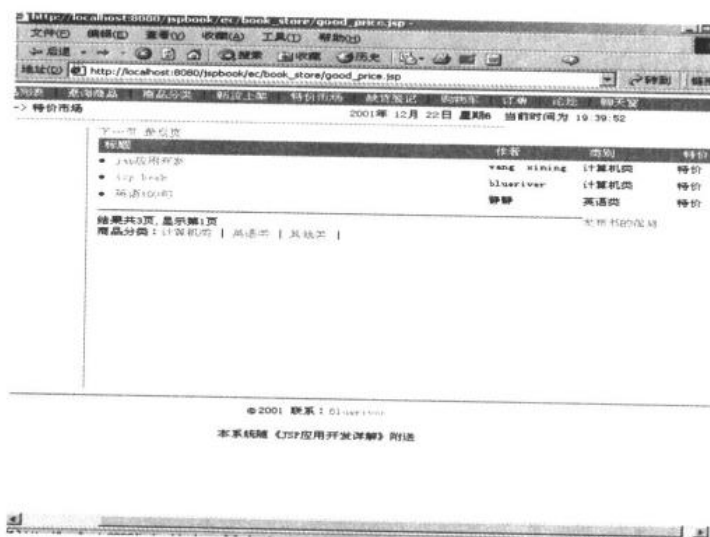


图 18-4 显示特价商品页面

在该特价商品的全部列表中，也采用了分页方式。本例子中，实现分页的原理是这样的：

(1) 首先，计算出满足条件的记录总数。

(2) 然后，纪录总数除以每页的显示个数。如果余数大于 0，那么逻辑页数应该为商+1；如果余数为 0，也就是说纪录总数整除以每页的显示个数，那么逻辑页数应为所得的商。

(3) 最后，根据请求的页数与实际的逻辑页数之间的关系进行读取数据记录。

当请求的页数值小于 1 时，就赋值给请求的页数为 1；同样地，当请求的页数值大于总的逻辑页数时，就赋值给请求的页数为总的逻辑页数值。

每一页中实际读取的记录数目应该是每页所显示的记录数，譬如在本例子中，其值应为 3。

文件 good\_goods.jsp 的源代码如例程 18-4 所示。

例程 18-4

```
//头文件 header.inc
<%@ include file="header.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<%! //自定义的静态函数，作用是：用数字分别来表示商品的种类。
public static String category(int id)
{
    try
    {
```

```

        switch(id)
        {
            case 1:
                return "计算机类";
            case 2:
                return "英语类";
            default:
                return "其他类";
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
            if(session.getAttribute("username")!=null){
            out.println(session.getAttribute("username"));
            }
            %> 当前位置: <a href="../index.jsp">首页</a> -&gt; 商品分类    </td>
            <!--显示时间--><%@ include file="../member/date.inc"%>    </tr>
            <TR bgColor=#3399ff>
                <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
                width=16></TD></TR>
                <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
                width=16></td></tr>
            </TBODY></TABLE>
            <table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
                <tr>
                    <td width="150" height="355" valign="top"></td>
                    <td width="10" height="100%"></td>
                    <td width="1" height="100%" bgcolor="#3399ff"></td>
                    <td width="10" height="100%"></td>
                    <td width="589" height="331" valign="top" background="images/bg1.gif">
                        <table border="0" width="100%" cellspacing="0" cellpadding="0">
                            <tr>
                                <td width="62%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff"> 标题
                                </font>
                                </td>
                                <td width="15%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff"> 作者
                                </font>

```



```

        </td>
        <td width="15%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff">类别
    </font>

        </td>
        <td width="8%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff">特价
    </font>

        </td>
    </tr>
    <ul>
<%////////算出共多少页
    int t;
    int mttotal;
    t=0;
    String strSQLsize="SELECT id FROM book where good_price=true";
    ResultSet Rssize = workM.executeQuery(strSQLsize);
    while(Rssize.next()){
        t=t+1;
    }
    //如果纪录总数除以每页的显示个数,余数大于 0,那么
    //逻辑页数应该为商+1;
    //否则, 逻辑页数应为所得的商
    //t 表示记录的总数
    //3 表示了每页所显示的个数
    //mttotal 为逻辑页数
    if((t%3)>0){
        mttotal=t/3+1;
    }else mttotal=t/3;
    %>
<%! //定义变量
String pageNo, mTmp;
    int i, j, k;
    %>
    <%
        //pageNo 表示请求的是第几页
        pageNo = request.getParameter("pageNo");
        //如果 pageNo 为 null 值, 则赋值为 1
        if(pageNo == null){
            pageNo = "1";
        }
        //j 表示的是 pageNo 对应的整型值
        j = Integer.parseInt(pageNo);
        //当 j 的值小于 1 时, 赋值为 1
    if(j < 1)

            j = 1;
        //当 j 的值大于总页数时, 赋值为 mttotal
        if(j > mttotal)

```

```

        j = mtotal;
    %>
    <% // 读取数据库记录
        String strSQL="SELECT * FROM book where good_price=true order by id desc";
        ResultSet RSa = workM.executeQuery(strSQL);
        //记录集移动到相应的位置
        //j 为逻辑页数
        for(k = 0;k < (j-1)*3;k++)
        {
            RSa.next();
        }
        i = 0;
        k = 1;
        while (RSa.next()) {
            i = i + 1;
            //超过 3 条，循环退出
            if(i == 4)
            {
                k = 0;
                break;
            }
            out.print("<tr height='23'><td><li><a href=javascript:show('book.jsp?id=");
out.print(RSa.getInt("id")+","'详细资料',480,500)");
            out.print(">" + getStr(RSa.getString("name")) + "</a></td><td>");
out.print(getStr(RSa.getString("author")) + "</td><td>");
out.print( (category(RSa.getInt("category")))+ "</td><td>特价</td></tr>");
            }
            i = i - k;
        //关闭记录集
        RSa.close();
    %>

    <%
    //以下代码行是用来显示页面数的
    //当 j 大于 1 时，就会显示“第一页”、“上一页”字样
    if(j > 1)
    {
    %>
        <a href="good_price.jsp?pageNo=1">第一页</a>
    <%
        int ii = Integer.parseInt(pageNo,10);
        if(ii > 1)
            ii = ii - 1;
        String ssTmp = Integer.toString(ii);
    %>
        <a href="good_price.jsp?pageNo=<%=ssTmp%>">上一页</a>
    <%

```

```

}
//同样地, 当 j 小于总逻辑页数时, 就会显示 “下一页”、“最后页” 字样
if(j < mttotal)
{
    int ii = Integer.parseInt(pageNo,10);
    if(ii < mttotal)
        ii = ii + 1;
    String ssTmp = Integer.toString(ii);
%>
    <a href="good_price.jsp?pageNo=<%=ssTmp%>">下一页</a>
    <a href="good_price.jsp?pageNo=<%=mttotal%>">最后页</a>
<%
}
if(mttotal < j)
    j = mttotal;
%>

<tr><td height="1" bgcolor="" colspan="2">
<hr color="#3399ff">
结果共<%=mttotal%>页,显示第<%=j%>页</td>
    <td align="left" colspan="2" height="32">
        <a href="post.jsp">发布书的信息</a>
    </td>
    <form name="form2" action="index.jsp" method="post">
        <input type="hidden" name="pageNo" value="">
    </form>
    <script language="javascript">
        function sub_page(id)
        {
            if(id==0)
                document.form2.pageNo.value=0;
            else if(id==1)
                document.form2.pageNo.value=2;
            document.form2.submit();
        }
    </script>
</tr>
</table>
</form>
商品分类: <a href="category.jsp?id=1">计算机类</a> |
    <a href="category.jsp?id=2">英语类 </a> |
    <a href="category.jsp?id=3">其他类 </a> |
</td> </tr> </table>
<!--尾文件-->
<%@ include file="footer.inc"%>

```

## 18.3 查看缺货商品列表

如果管理者想要查看缺货商品的列表，以便及时地去采购或者以其他方式来补充缺货商品，那么可以通过单击导航栏上的缺货登记来查看缺货商品的全部列表，如图 18-5 所示。

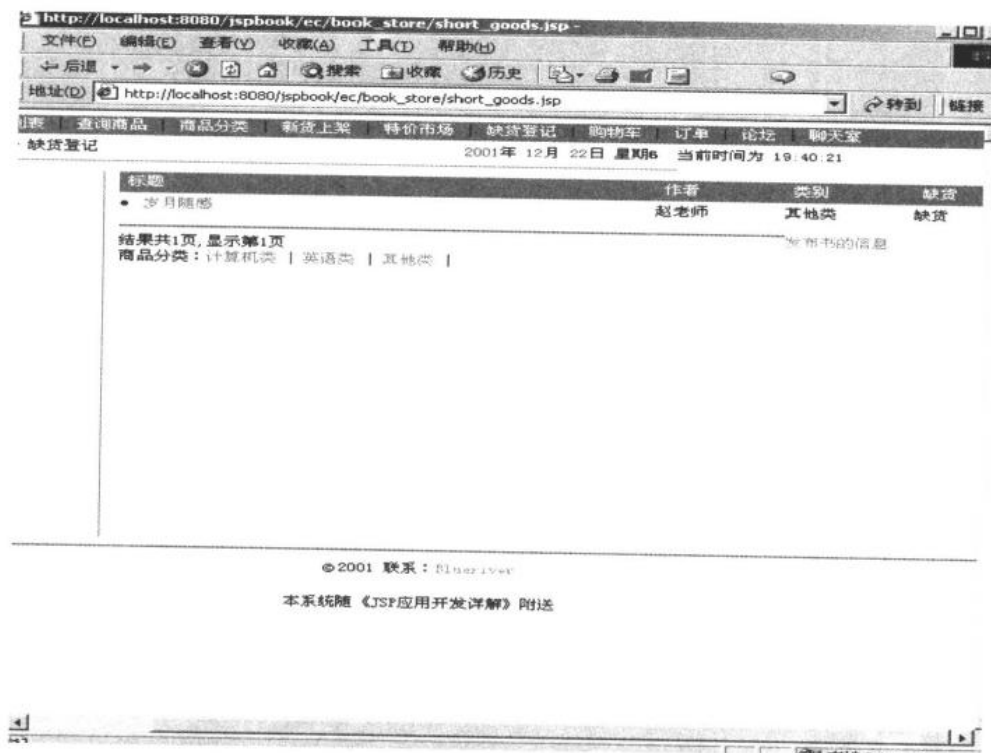


图 18-5 缺货商品的列表

在该缺货商品的全部列表中，也采用了分页方式。

文件 `short_goods.jsp` 的源代码在这里就不再列出，请参考附带光盘中的文件。



# 第 19 章 购 物 车

## 1. 选购商品放入购物车页面及其源代码

登录用户可以选择想要购买的商品，然后放入自己的购物车，这很像实际的超市购物一样。如图 19-1 所示商品的详细资料列表，单击放入购物车链接，则会将该商品放入用户的购物车中。

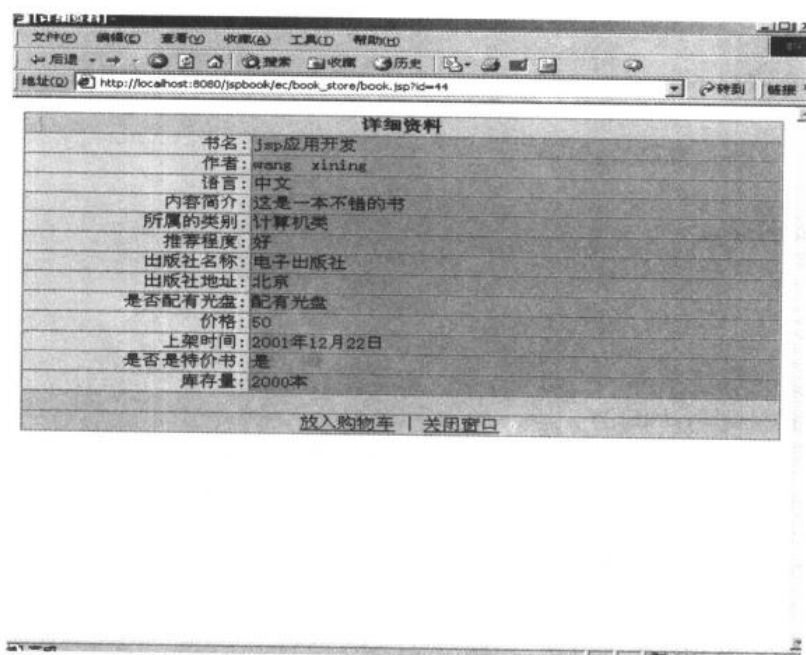


图 19-1 显示商品

图 19-1 所示的界面中，列出了所选中的商品的详细介绍。在该页面中，有两个链接——一个是“放入购物车”链接，单击该链接，则会将商品放入购物车；另外一个链接则是用来关闭窗口。该页面所对应的文件是 `cart.jsp`，源代码如例程 19-1 所示。

例程 19-1

```
//导入必要的包 java.sql.*
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.jsp"%>
<%//判断 session (用户) 是否有效
if(session.getAttribute("username")==null||session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
```



```

    }
    %>
    //处理中文问题的自定义函数
    <%!
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("GBK");
            String temp=new String(temp_t,"ISO8859_1");
            return temp;
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        return "null";
    }
    %>
    <html>
    <head>
    <title>[详细资料]</title>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    </head>
    <body bgcolor="#FFFFFF">
    <div align="center">
        <table width="100%" border="0" bgcolor="#CCCCFF">
            <tr>
                <td>
                    <div align="center"><b><font color="#FF0000">购 物 车</font></b></div>
                </td>
            </tr>
        </table>
        <form name="form1" method="post" action="order1.jsp">
        <p>您的购物车中包含以下货物: </p>
        <table width="100%" border="0" align="center">
            <tr bgcolor="#FFCCCC">
                <td width="15%">
                    <div align="center"><font color="#0000FF">购买数量</font></div>
                </td>
                <td width="56%">
                    <div align="center"><font color="#0000FF">书名</font></div>
                </td>
                <td width="13%">

```

```

        <div align="center"><font color="#0000FF">单价</font></div>
    </td>
    <td width="16%">
        <div align="center"><font color="#0000FF">总价格</font></div>
    </td>
</tr>
<%
//定义变量（价格）并进行初始化
double g_price,total_price;
g_price=0;
total_price=0;
//执行数据库操作，查出还没有进行处理过的当前用户的订单
//status=0 表示订单还没有被处理过
//status=1 则表示订单已经被处理过
String sqlList="select * from orders where user_name="";
sqlList = sqlList +session.getAttribute("username")+"" and status=0";
ResultSet RSList=workM.executeQuery(sqlList);
try
{
    while(RSList.next())
    {
        int b_num;
        b_num=RSList.getInt("book_number");
%>
<tr bgcolor="#CCFFCC">
    <td width="15%">
        <div align="center"> <font color="#0000FF"> <%=b_num%>
            <input type="hidden" name="book_number" size="4" value=<%=b_num%>>
        </font></div>
    </td>
    <%
        String sqlBook="select book.* from book where id="+RSList.getInt("book_id");
        ResultSet RSBook=workM.executeQuery(sqlBook);
        while(RSBook.next())
        {
            %>
            <td width="56%">
                <div align="center"><font color="#0000FF">
                    <%=getStr(RSBook.getString("name"))%></font></div>
            </td>
            <td width="13%">
                <%
                    double price;
                    price=RSBook.getDouble("price");
                %>
                <div align="center"><font color="#0000FF">&yen;<%=price%></font></div>
            </td>
        }
    }
}
catch (SQLException e)
{
    %>
    <td colspan="3"><font color="#FF0000">
        <%=e.getMessage()%></font></td>
}
finally
{
    %>
    workM.close();
}
%>

```

434

```

        <input type="button" value="继续购物" onclick="javascript:self.close();">
        <input type="submit" name="Submit2" value="填写订单">
    </td>
</tr>
</table>
</form>
<p>&nbsp;</p>
</div>
</body>
</html>

```

在图 19-2 所示，页面中显示了当前用户还没有处理过的订单。在例子中，可以看到当前用户 **blueriver** 还没有被处理的订单中，包含有三种商品（书）。

在图 19-2 所示的页面中，详细列出了订单中商品的数量、商品单价、每种商品的总价格以及总共的货物价格，另外还显示了运输费用（在本例子中，我们固定了运输费用为 5 元，读者可以在实际的应用中做相应的修改）。最后，在页面的最下端，则显示了该订单的总费用。

单击【继续购物】按钮，则可以继续购物；单击【填写订单】按钮，则会出现让用户填写详细送货资料的页面。

购物车			
您的购物车中包含以下货物:			
购买数量	书名	单价	总价格
1	岁月随感	¥120.1	¥120.1
1	齐秦歌词大全	¥100.0	¥100.0
1	jsp/java Grammar Book	¥123.2	¥123.2
货物价格			¥343.3
运输费用			¥5.00
总费用			¥348.3
继续购物		填写订单	

图 19-2 购物车页面

## 2. 显示购物车中商品信息

用户选购完所需的商品之后，需要填写个人资料，以便商品提供商能够及时准确地送货给用户，如图 19-3 所示。

如果用户想继续购物，那么可以单击【继续购物】按钮；如果购物完毕，则需要填写完个人资料并提交订单。在实际中，需要填写的个人资料应该是非常详细的，在这个例子中，笔者只是列举了地址和电话。

请您详细填写以下资料，然后单击“发出订单”按钮

姓名:	blueriver
住址:	北京海淀
电话:	12345678

您的购物车中包含以下货物:

购买数量	书名	单价	总价格
1	岁月随感	¥120.1	¥120.1
1	齐秦歌词大全	¥100.0	¥100.0
1	jsp/java Grammar Book	¥123.2	¥123.2
货物价格			¥343.3
运输费用			¥5.00
总费用			¥348.3

图 19-3 填写订单

文件 order1.jsp 的源代码如例程 19-2 所示。

例程 19-2

```
//导入必要的包 java.sql.*
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.jsp"%>
<%//判断 session (用户) 是否有效
if(session.getAttribute("username")==null||session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
<%! //处理中文问题的自定义函数

public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
```

```

%>
<html>
<head>
<title>[详细资料]</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body bgcolor="#FFFFFF">
.....部分 HTML 代码省略（详细代码请参看附带光盘）
    <td width="12%" bgcolor="#FFCCCC">姓名:</td>
    <td width="88%" bgcolor="#CCFFCC"><%=session.getAttribute("username")%></td>
</tr>
<tr>
    <td width="12%" bgcolor="#FFCCCC">住址:</td>
    <td width="88%" bgcolor="#CCFFCC">
        <input type="text" name="address">
    </td>
</tr>
<tr>
    <td width="12%" bgcolor="#FFCCCC">电话:</td>
    <td width="88%" bgcolor="#CCFFCC">
        <input type="text" name="tel">
    </td>
</tr>
<tr>
    <td colspan=2>
        <div align="center">
            <input type="submit" name="Submit2" value="提交订单">
            <input type="button" value="继续购物" onClick="javascript:self.close();" name="button">
        </div>
    </td>
</tr>
</table>
</form>
<p>您的购物车中包含以下货物:</p>
<table width="100%" border="0" align="center">
    <tr bgcolor="#FFCCCC">
        <td width="15%">
            <div align="center"><font color="#0000FF">购买数量</font></div>
        </td>
        <td width="56%">
            <div align="center"><font color="#0000FF">书名</font></div>
        </td>
        <td width="13%">
            <div align="center"><font color="#0000FF">单价</font></div>

```



```

        </td>
        <td width="16%">
            <div align="center"><font color="#0000FF">总价格</font></div>
        </td>
    </tr>
    <%
        //价格
        double g_price,total_price;
        g_price=0;
        total_price=0;
        //查询还没有被处理的订单资料
        String sqlList="select * from orders where user_name=";
        sqlList = sqlList +session.getAttribute("username")+"" and status=0";
        ResultSet RSList=workM.executeQuery(sqlList);
        try
        {
            while(RSList.next())
            {
                int b_num;
                b_num=RSList.getInt("book_number");
            }
        }
    %>
    <tr bgcolor="#CCFFCC">
        <td width="15%">
            <div align="center"> <font color="#0000FF"> <%=b_num%>
                <input type="hidden" name="book_number" size="4" value=<%=b_num%>>
            </font></div>
        </td>
        <%
            String sqlBook="select book.* from book where id="+RSList.getInt("book_id");
            ResultSet RSBook=workM.executeQuery(sqlBook);
            while(RSBook.next())
            {
                %>
                <td width="56%">
                    <div align="center"><font color="#0000FF">
                        <%=getStr(RSBook.getString("name"))%></font></div>
                    </td>
                    <td width="13%">
                        <%
                            double price;
                            price=RSBook.getDouble("price");
                        %>
                        .....部分 HTML 代码省略（详细代码请参看附带光盘）
                    </td>
                </td>
            }
        %>
    </tr>
</body>
</html>

```

# 第 20 章 用 户 订 单

“用户订单”是电子商务中最常用的网页技术，下面我们来详细讲解它的实现过程。

## 20.1 订单信息确认

用户填写完订单所需的个人资料后，就可以提交订单了。如图 20-1 所示的页面就是一张用户 **blueriver** 的订单信息确认单，也就是一张订单通知单。

该订单通知中，显示了订单用户的详细个人资料以及用户购物车中的商品价格、数量以及运输费用、总费用等。

用户可以将订单打印保存，以便与送货人确认。

订单信息确认

订单信息

姓名:	blueriver
住址:	北京海淀
电话:	12345678

您的购物车中包含以下货物:

购买数量	书名	单价	总价格
1	岁月随感	¥120.1	¥120.1
1	齐秦歌词大全	¥100.0	¥100.0
1	jsp/java Grammar Book	¥123.2	¥123.2
货物价格			¥343.3
运输费用			¥5.00
总费用			¥348.3

关闭窗口 | 打印该页

图 20-1 订单信息通知

文件 `submit_order.jsp` 源代码如例程 20-1 所示。

例程 20-1

```
//导入必要的包 java.sql.*
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.jsp"%>
```

```
<%//判断 session（用户）是否有效
if(session.getAttribute("username")==null||session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
<%!//处理中文问题的自定义函数
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<%!//处理中文问题的自定义函数
public String getStr2(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<% //定义变量，用来保存用户的个人资料
String t_address,t_tel;
String address,tel;
t_address=request.getParameter("address");
t_tel=request.getParameter("tel");
//进行中文处理
```

```
address=getStr(t_address);
tel=getStr(t_tel);
//进行数据库操作
String sqlinsert="update orders set user_address='"+address+"',user_tel='"+tel;
sqlinsert = sqlinsert + " where user_name='"+session.getAttribute("username")+"'";
workM.executeQuery(sqlinsert);
%>
.....部分 HTML 代码省略（详细代码请参看附带光盘）
<%
//价格
double g_price,total_price;
g_price=0;
total_price=0;
String sqlList="select * from orders where user_name='";
sqlList= sqlList+session.getAttribute("username")+"' and status=0";
//执行数据库操作
ResultSet RSList=workM.executeQuery(sqlList);
try
{
    while(RSList.next())
    {
        int b_num;
        b_num=RSList.getInt("book_number");
    }
}
%>
.....部分 HTML 代码省略（详细代码请参看附带光盘）
```

## 20.2 列出所有未处理的订单列表

为了管理员能够方便地查看所有的未处理的订单列表，开发人员需要提供一个显示所有未处理订单的页面，如图 20-2 所示。

在图 20-2 所示的页面中，显示了订单号、订单用户名以及订单用户的其他个人资料。

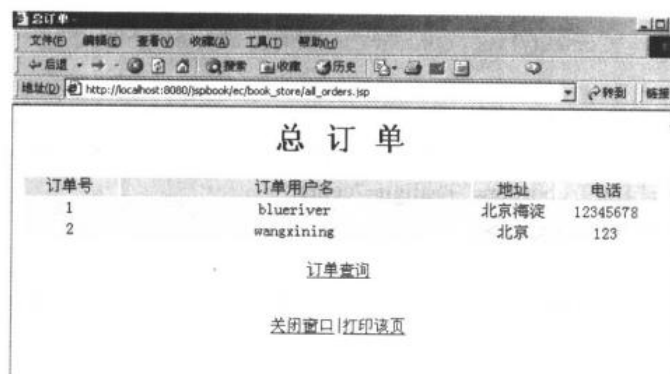


图 20-2 订单列表

文件 all\_orders.jsp 的源代码如例程 20-2 所示。

例程 20-2

```
//导入必要的包 java.sql.*
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.jsp"%>
<%//判断 session (用户) 是否有效
if(session.getAttribute("username")==null||session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
<%! //处理中文问题的自定义函数
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<html>
<head>
<title>总订单</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body bgcolor="#FFFFFF">
    <p><b><div align="center"><font size="+3">总 订 单</font></div></b></p>
    <table width="100%" border="0" align="center">
        <tr bgcolor="#FFCCCC">
            <td width="15%">
                <div align="center"><font color="#0000FF">订单号</font></div>
            </td>
            <td width="56%">
                <div align="center"><font color="#0000FF">订单用户名</font></div>
            </td>
        </tr>
    </table>
</body>
</html>
```

```

        <td width="13%">
            <div align="center"><font color="#0000FF">地址</font></div>
        </td>
        <td width="16%">
            <div align="center"><font color="#0000FF">电话</font></div>
        </td>
    </tr>
    <%
//执行数据库操作，查询所有的未处理的订单
String sql;
sql="select distinct user_name,user_address,user_tel from orders where status=0";
ResultSet RS=workM.executeQuery(sql);
try
{
    int i=1;
    while(RS.next())
    {
        String user_name,user_address,user_tel;
        user_name=RS.getString("user_name");
        user_address=RS.getString("user_address");
        user_tel=RS.getString("user_tel");

        %>
        <tr bgcolor="#CCFFCC">
            <td width="15%">
                <div align="center"> <font color="#0000FF"> <%=i%>
                </font></div>
            </td>
            <td width="56%">
                <div align="center"><font color="#0000FF"><%=getStr(user_name)%></font></div>
            </td>
            <td width="13%">
                <div align="center"><font color="#0000FF">
        <%=getStr(user_address)%></font></div>
            </td>
            <td width="16%">
                <div align="center"><font color="#0000FF"><%=getStr(user_tel)%></font></div>
            </td>
        <%
            i=i+1;
        }
    }
    catch(Exception ex)
    {
        out.println(ex.getMessage());
    }
    %>

```



```
<tr align="center">
  <td colspan="4">
    <br><a href="search_order1.jsp">订单查询</a>
  </td>
</tr>
<tr align="center">
  <td colspan="4">
    <br><br>
    <a href="javascript:self.close();">关闭窗口</a>|<a href="javascript:self.print();">打印该页</a>
  </td>
</tr>
</table>
<p>&nbsp;</p>
</div>
</body>
</html>
```

## 20.3 查询订单

如果要查询特定用户的订单，则可以通过如图 20-3 所示的页面进行查询。

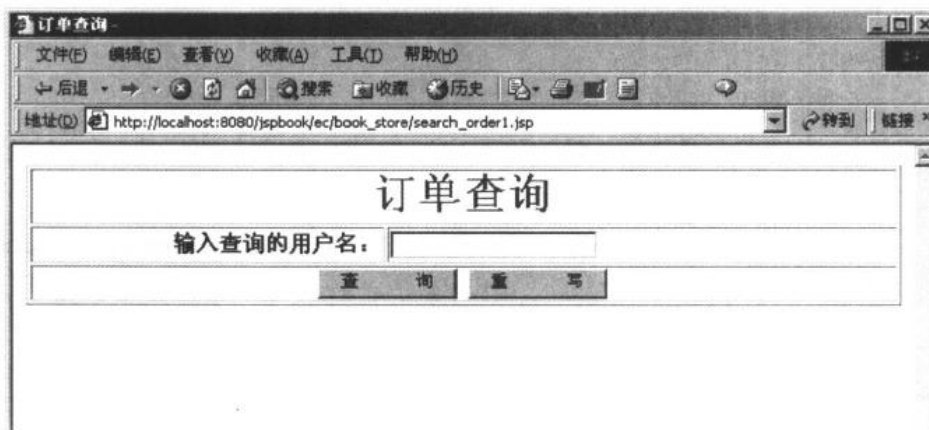


图 20-3 订单查询

文件 search\_order1.jsp 的源代码如例程 20-3 所示。

例程 20-3

```
<html>
<head>
<title> 订单查询</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body bgcolor="#FFFFFF">
<form method="post" action="search_order2.jsp">
```

.....部分 HTML 代码省略（详细代码请参看附带光盘）

```
</form>
</body>
</html>
```

输入要查询的用户名后，单击【查询】按钮进行查询，将会得到如图 20-4 所示的界面。在图 20-4 的页面中，显示了订单用户的个人资料以及订单的详细资料。

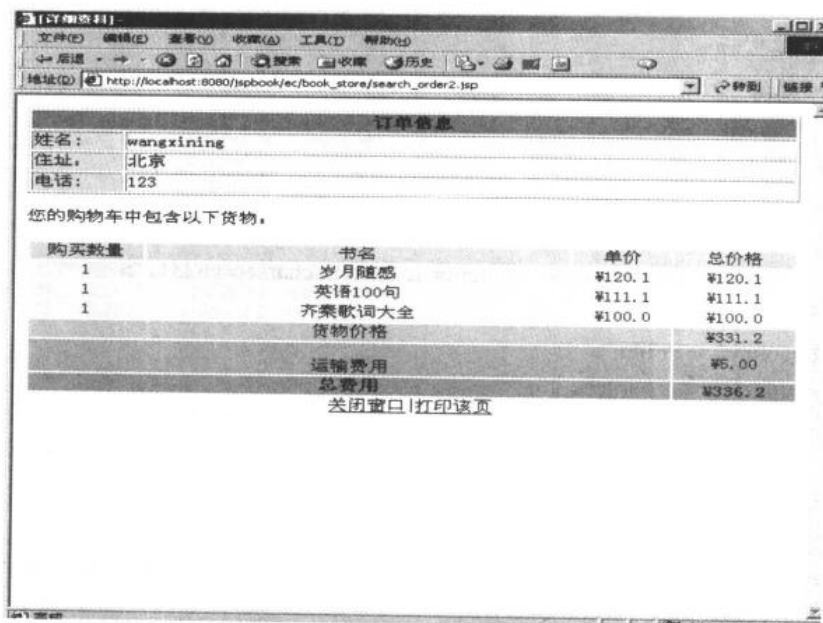


图 20-4 订单查询结果

文件 search\_order2.jsp 的源代码如例程 20-4 所示。

例程 20-4

```
//导入必要的包 java.sql.*
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
<%@ include file="function.inc.jsp"%>
<%//判断 session (用户) 是否有效
if(session.getAttribute("username")==null||session.getAttribute("username")== "")
{
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
<%! //处理中文问题的自定义函数
public String getStr(String str)
{
    try
    {
        String temp_p=str;
```

```

        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<html>
<head>
<title>[详细资料]</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body bgcolor="#FFFFFF">
<table width="100%" border="1">
    .....部分 HTML 代码省略（详细代码请参看附带光盘）
<%
//进行数据库操作， 查询特定用户的订单
String address,tel;
String sql="select * from orders where user_name='"+request.getParameter("username")+"'";
ResultSet rs=workM.executeQuery(sql);
if(rs.next())
{
    address=getStr(rs.getString("user_address"));
    tel=getStr(rs.getString("user_tel"));
}
%>
    .....部分 HTML 代码省略（详细代码请参看附带光盘）
<% }%>
</table>
</form>
<p>您的购物车中包含以下货物: </p>
    .....部分 HTML 代码省略（详细代码请参看附带光盘）
<%
//定义变量，用来保存商品的价格
double g_price,total_price;
g_price=0;
total_price=0;
//进行数据库操作， 查询订单中的商品以及订单的详细资料
String sqlList="select * from orders where user_name='";
sqlList = sqlList +request.getParameter("username")+"' and status=0";
ResultSet RSList=workM.executeQuery(sqlList);
try
{

```

```
        while(RSList.next())
        {
            int b_num;
            b_num=RSList.getInt("book_number");
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
<%>
.....部分代码省略（详细代码请参看附带光盘）
<tr align="center">
    <td colspan="4">
        <a href="javascript:self.close();">关闭窗口 </a><a href="javascript:self.print();">打印该页
    </a>
    </td>
</tr>
</table>
<p>&nbsp;</p>
</div>
</body>
</html>
```



# 第 21 章 论 坛

## 1. 说明

在该例子中，论坛的用户还是使用前面在线超市系统的用户。有关用户的表结构请参看第 14 章中的介绍。

另外，在本例子中还用到了其他的数据库。所用的数据库表结构如下。

表 topic 用来存储主题帖子的信息，其结构如图 21-1 所示。

图 topic: 表			
	字段名称	数据类型	
▶	id	自动编号	
	title	文本	帖子的标题
	author	文本	帖子的作者
	email	文本	帖子的作者的EMAIL
	content	备注	帖子的内容
	date	日期/时间	发帖子的时间

图 21-1 表 topic 的结构

表 reply 用来存储回复帖子的信息，其结构如图 21-2 所示。

图 reply: 表			
	字段名称	数据类型	
▶	id	自动编号	
	topicID	数字	主题贴子的id
	title	文本	回复帖子的标题
	author	文本	回复帖子的作者
	email	文本	回复帖子的作者的EMAIL
	content	备注	回复帖子的内容
	date	日期/时间	回复帖子的时间

图 21-2 表 reply 的结构

## 2. 论坛主帖子的列表效果图及其源代码

在论坛帖子的列表中，用到了分页系统。在下面的代码中，分别用黑体字注释了实现分页功能的代码。

在实际的项目开发中，分页系统有着非常广泛的应用。笔者希望读者，通过对这个例子的理解，能够真正理解实现分页的原理，从而能够在实际的应用中加以推广。

本例子中，实现分页的原理是这样的：

- 首先，计算出满足条件的记录总数。



- 然后, 纪录总数除以每页的显示个数。如果余数大于 0, 那么逻辑页数应该为商+1;

如果余数为 0, 也就是说纪录总数整除每页的显示个数, 那么逻辑页数应为所得的商。

- 最后, 根据请求的页数与实际的逻辑页数之间的关系进行读取数据记录。

当请求的页数值小于 1 时, 就赋值给请求的页数为 1; 同样地, 当请求的页数值大于总的逻辑页数时, 就赋值给请求的页数为总的逻辑页数值。

每一页中实际读取的记录数目应该是每页所显示的记录数, 譬如在本例子中, 其值应为 10, 最后执行效果如图 21-3 所示。

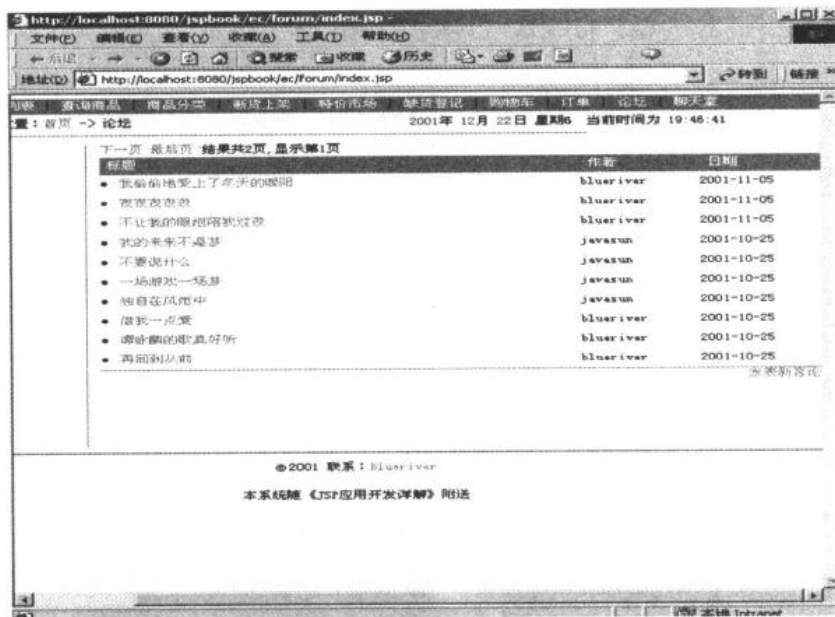


图 21-3 主帖子列表

forum/index.jsp 文件的源代码如例程 21-1 所示。

#### 例程 21-1

```
<!--包含的头文件-->
<%@ include file="../book_store/head.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
```

```

        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>

<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>
      <% //显示用户名
      if(session.getAttribute("username")!=null){
        out.println(session.getAttribute("username"));
      }
      %> 当前位置: <a href="../index.jsp">首页</a> -&gt; 成员服务 </td>
      <td align="right">
        <!--显示时间-->
        <%@ include file="../member/date.inc"%>
      </td>
      .....部分代码省略（详细代码请参看附带光盘）
      <%////////算出共多少页
        int t;
        int mttotal;
        t=0;
        String strSQLsize="SELECT id FROM topic";
        ResultSet RSsize = workM.executeQuery(strSQLsize);
        while(RSsize.next()){
          t=t+1;
        }
        //如果纪录总数除以每页的显示个数,余数大于 0,那么
        //逻辑页数应该为商+1;
        //否则, 逻辑页数应为所得的商
        //t 表示记录的总数
        //10 表示了每页所显示的个数
        //mttotal 为逻辑页数
        if((t%10)>0){
          mttotal=t/10+1;
        }else mttotal=t/10;
      %>
      <%!String pageNo, mTmp;

```

```
int i, j, k;
%>
<%
    //pageNo 表示请求的是第几页
    pageNo = request.getParameter("pageNo");
    //如果 pageNo 为 null 值, 则赋值为 1
    if(pageNo == null){
        pageNo = "1";
    }
    //j 表示的是 pageNo 对应的整型值
    j = Integer.parseInt(pageNo);
    //当 j 的值小于 1 时, 赋值为 1
    if(j < 1)
        j = 1;
    //当 j 的值大于总页数时, 赋值为 mttotal
    if(j > mttotal)
        j = mttotal;
%>
<% // 读取数据库记录
    String strSQL="SELECT * FROM topic order by id desc";
    ResultSet RSa = workM.executeQuery(strSQL);
    //记录集移动到相应的位置
    //j 为逻辑页数
    for(k = 0;k < (j-1)*10;k++)
    {
        RSa.next();
    }
    i = 0;
    k = 1;
    while (RSa.next()) {
        i = i + 1;
        //超过 10 条, 循环退出
        if(i == 11)
        {
            k = 0;
            break;
        }
        out.print("<tr height='23'><td><li><a href=article.jsp?id="+RSa.getInt("id");
        out.print(">" + getStr(RSa.getString("title")) + "</a></td><td>");
        out.print(getStr(RSa.getString("author")) + "</td><td>");
        out.print(RSa.getDate("date") + "</td></tr>");
    }
    i = i - k;
    //关闭记录集
    RSa.close();
%>
```

```

<%
//以下代码行是用来显示页面数的
//当 j 大于 1 时, 就会显示“第一页”、“上一页”字样
if(j > 1)
{
%>
    <a href="index.jsp?pageNo=1">第一页</a>
<%
    int ii = Integer.parseInt(pageNo,10);
    if(ii > 1)
        ii = ii - 1;
    String ssTmp = Integer.toString(ii);
%>
    <a href="index.jsp?pageNo=<%=ssTmp%>">上一页</a>
<%
}
//同样地, 当 j 小于总逻辑页数时, 就会显示“下一页”、“最后页”字样
if(j < mttotal)
{
    int ii = Integer.parseInt(pageNo,10);
    if(ii < mttotal)
        ii = ii + 1;
    String ssTmp = Integer.toString(ii);
%>
    <a href="index.jsp?pageNo=<%=ssTmp%>">下一页</a>
    <a href="index.jsp?pageNo=<%=mttotal%>">最后页</a>
<%
}
if(mttotal < j)
    j = mttotal;
%>
结果共<%=mttotal%>页,显示第<%=j%>页
.....部分代码省略(详细代码请参看附带光盘)
</td>
</tr>
</table>
<!--尾文件-->
<%@ include file="../member/footer.inc"%>

```

### 3. 查看每条帖子的详细情况

在帖子的列表中, 单击帖子标题, 则可以查看每条帖子的具体内容以及回复该帖子的答复内容等。如图 21-4 所示。

在如图 21-4 所示的页面中, 最上端的帖子为主帖子, 跟在主帖子下面的帖子是主帖子的回复内容。

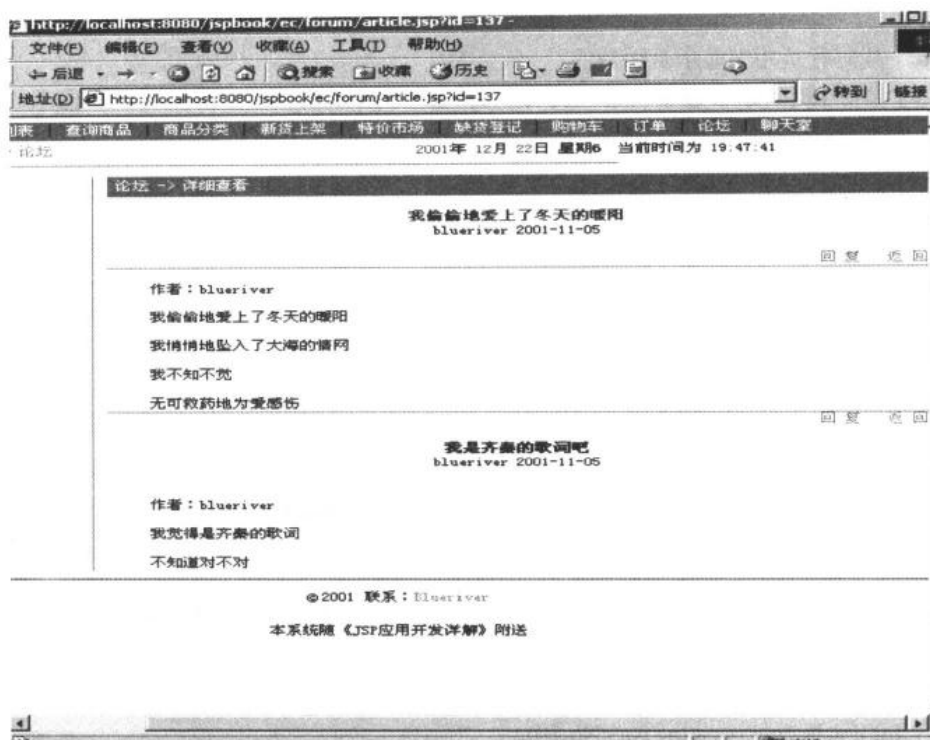


图 21-4 帖子及其回复

文件 article.jsp 的源代码如例程 21-2 所示。

例程 21-2

```
<!--包含的头文件-->
<%@ include file="../book_store/head.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
}
```

```

    {
        e.printStackTrace();
    }
    return "null";
}
%>
.....部分代码省略（详细代码请参看附带光盘）
<% //参数 id 表示帖子对应的 id
    String id;
    id=request.getParameter("id");
    String strSQL="SELECT * FROM topic where id="+id;
    //执行数据库操作， 查询帖子
    ResultSet RSa = workM.executeQuery(strSQL);
    while (RSa.next()) {
        String a_author;
        int topic_id;
        topic_id=RSa.getInt("id");
        a_author=getStr(RSa.getString("author"));
        out.print("<br><b><font color=red>"+getStr(RSa.getString("title"));
out.print("</font></b><br>"+a_author+" "+RSa.getDate("date")+ "<br><br>");
out.print("</td></tr><tr><td colspan='3' align='right'>");
    //判断帖子的作者是否是 session.getValue("username");
    //如果是，那么则可以出现编辑、删除该帖子的功能；
    //相反，如果不是，则用户不能编辑或者删除该帖子。
    if(a_author.equals(session.getValue("username")))
    {
        out.print("<a href=edit.jsp?mode=topic&id="+topic_id+">编辑</a>&nbsp;&nbsp;&nbsp;");
out.print("<a href=delete.jsp?mode=topic&id="+topic_id+">删除</a> &nbsp;&nbsp;&nbsp;");
    }
    // 任何成功登录的用户都可以回复帖子
    out.print("<a href=reply.jsp?id="+id+">回 复</a> &nbsp;&nbsp;&nbsp;");
out.print("<a href=index.jsp>返 回</a></td></tr><tr><td colspan='3'");
out.print(" height='1' bgcolor=#3399ff></td></tr><tr><td width='5%'></td>");
    out.print("<td><br>作者: "+a_author+"<br><br>");
    out.print(getStr(RSa.getString("content")));
}
RSa.close();
%>

</td>
<td width="5%"></td>
</tr>
<% //查询主帖子的回复帖子
    String strRe="SELECT * FROM reply where topicID="+id;
    ResultSet RSr = workM.executeQuery(strRe);

```



```

while (RSr.next()) {
    String a_author,t_id;
    t_id=RSr.getString("id");
    a_author=getStr(RSr.getString("author"));
    out.println("<tr><td colspan='3' height='1' bgcolor='#3399ff'></td></tr>");
    out.println("<tr><td colspan='3' align='right'>");
    //判断帖子的作者是否是 session.getValue("username");
    //如果是，那么则可以出现编辑、删除该帖子的功能；
    //相反，如果不是，则用户不能编辑或者删除该帖子。
    if(a_author.equals(session.getValue("username")))
    {
        out.print("<a href=edit.jsp?mode=reply&id="+t_id+">编 辑</a>");
    out.print("&nbsp;&nbsp;&nbsp;<a href=delete.jsp?mode=reply&id="+t_id);
    out.print(">删 除</a> &nbsp;&nbsp;&nbsp;");
    }
    // 任何成功登录的用户都可以回复帖子
    out.print("<a href=reply.jsp?id="+id+">回 复</a>");
    out.print("&nbsp;&nbsp;&nbsp;<a href=index.jsp>返 回</a></td></tr>");
    out.println("<tr><td align=center colspan='3' valign=top>");
    out.print("<br><b>"+getStr(RSr.getString("title"))+"</b><br>");
    out.print("<a author=" + a_author + " " + RSr.getDate("date") + "<br><br></td></tr>");
    out.print("<tr><td width='5%'></td>");
    out.print("<td><br>作者: " + a_author + "<br><br>");
    out.print(getStr(RSr.getString("content")));

    %>
    </td>
    <td width="5%"></td>
    </tr>

    <%
    }
    //关闭数据库链接
    RSr.close();
    %>
    </table>
    </td>
    </tr>
    </table>
    <!--包含尾文件-->
    <%@ include file="../member/footer.inc"%>

```

#### 4. 回复帖子

用户对于每一个帖子都可以进行回复，回复帖子的功能实现与用户发表新贴子的功能实现在原理上基本相同，如图 21-5 所示。

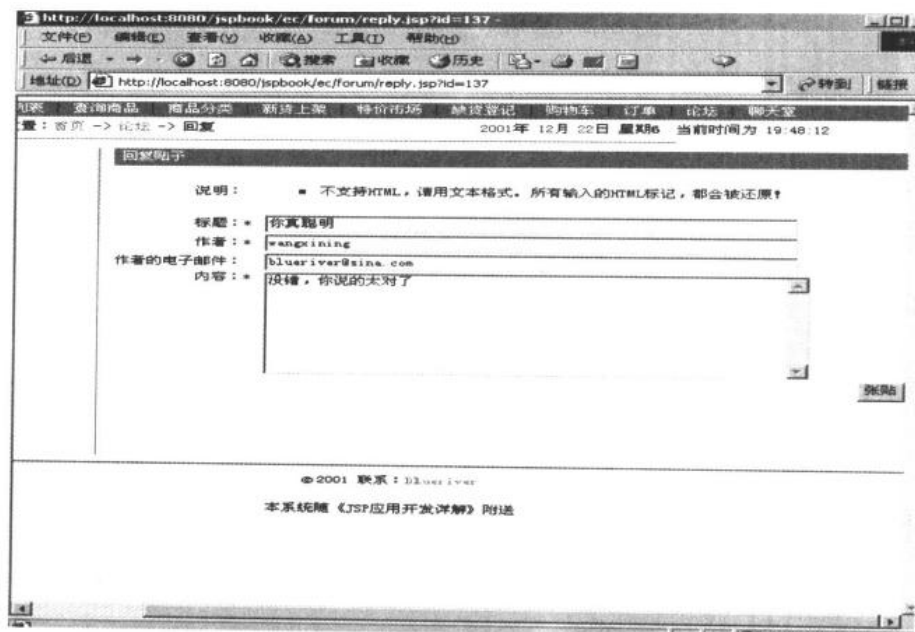


图 21-5 回复帖子

文件 reply.jsp 的源代码如例程 21-3 所示。

例程 21-3

```
<% //判断用户是否已经成功登录;
    //如果没有登录成功, 页面则会导向登录界面。
if(session.getAttribute("username")==null){
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
<!--包含的头文件-->
<%@ include file="../book_store/head.inc"%>
//调用用来显示时间的 JspBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type='dates.JspCalendar' /> <TABLE
border=0 cellpadding=0 cellspacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
            if(session.getAttribute("username")!=null){
                out.println(session.getAttribute("username"));
            }
            %> 当前位置: <a href="../index.jsp">首页</a> -&gt; 成员服务 </td>
            <td align="right"><!--显示时间--> <%@ include file="date.inc"%> </td>
            .....部分代码省略(详细代码请参看附带光盘)
            <form name="form1" method="post" action="reply_ok.jsp">
                <input type="hidden" name="topicID" value="<%=request.getParameter("id")%>">
                <table width="100%" border="0" cellspacing="0" cellpadding="0">
                    <tr><td colspan="3" height="20"></td></tr>
                    <tr>

```

.....部分代码省略（详细代码请参看附带光盘）

```

<tr>
  <td align="right">标题: </td>
  <td width="20">*</td>
  <td width="500">
    <input type="text" name="title" size="64" value="">
  </td>
</tr>
<tr>
  <td align="right">作者: </td>
  <td width="20">*</td>
  <td width="500">
    <input type="text" name="author" size="64"
value="<%=session.getValue("username")%>" readonly>
  </td>
</tr>
<tr>
  <td align="right">作者的电子邮件: </td>
  <td width="20">&nbsp;</td>
  <td width="500">
    <input type="text" name="aemail" size="64" value="">
  </td>
</tr>
<tr>
  <td align="right" valign="top">内容: </td>
  <td width="20" valign="top">*</td>
  <td>
    <textarea name="content" cols="64" rows="8"></textarea>
  </td>
</tr>
<tr>
  <td align="right">&nbsp;</td>
  <td width="20">&nbsp;</td>
  <td align="right">
    <input type="button" name="post" value=" 张 贴 "
onclick="sub()">
  </td>
</tr>
</table>
</form>
</td>
</tr>
</table>
<script language="javascript">
function sub()
{

```

```

        if(document.form1.title.value=="")
        {
            window.alert("请输入文章标题! ");
            document.form1.title.focus();
            return false;
        }
        if(document.form1.author.value=="")
        {
            window.alert("请输入作者姓名! ");
            document.form1.author.focus();
            return false;
        }
        if(document.form1.content.value=="")
        {
            window.alert("请输入文章内容! ");
            document.form1.content.focus();
            return false;
        }
        document.form1.submit();
    }
</script>
</td>
</tr>
</table>
<!--包含的尾文件-->
<%%@ include file="../member/footer.inc"%>

```

### 5. 回复帖子逻辑处理源代码

输入完回复帖子的内容，提交表单信息，那么回复帖子的逻辑处理过程将在文件 `reply_ok.jsp` 中进行。

在 `reply_ok.jsp` 文件中，分别定义了两个函数：一个函数名为 `returnToBr()`，用来转换字符串中的空格、回车字符；另一个函数名为 `returnToHTML()`，用来转换字符串中的 HTML 字符。

其源代码如例程 21-4 所示。

例程 21-4

```

//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try

```

```

    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>
<%! //处理输入的信息内容
    //该函数能够将字符串 sStr 中的'\n'、'\r'转换为<br>;
    //也能够将字符串 sStr 中的' '转换为&nbsp;;
    public static String returnToBr(String sStr)
    //如果 sStr 为 null 或者 sStr 为空值;
    //则该函数会返回不发生变化的 sStr;
    if (sStr == null || sStr.equals(""))
    return sStr;
    }
    //定义 StringBuffer 型的变量 sTmp
    StringBuffer sTmp = new StringBuffer();
    int i = 0;
    while (i <= sStr.length()-1)
    //将字符串 sStr 中的'\n'、'\r'转换为<br>;
    if (sStr.charAt(i) == '\n' || sStr.charAt(i) == '\r')
    sTmp = sTmp.append("<br>");
    } else if (sStr.charAt(i) == ' ')
    将字符串 sStr 中的' '转换为&nbsp;;
    sTmp = sTmp.append("&nbsp;");
    }else
    sTmp = sTmp.append(sStr.substring(i,i+1));
    }
    i++;
    }
    String S1;
    S1=sTmp.toString();
    return S1;
    }
%>
<%! //处理输入的信息内容
    //该函数能够将字符串 sStr 中的'<' 转换为"&lt;";
    //也能够将字符串 sStr 中的'>' 转换为"&gt;";
    public static String returnToHTML(String sStr)

```

```

//如果 sStr 为 null 或者 sStr 为空值;
//则该函数会返回不发生变化的 sStr;
if (sStr == null || sStr.equals(""))
return sStr;
}
//定义 StringBuffer 型的变量 sTmp
StringBuffer sTmp1 = new StringBuffer();
int i = 0;
while (i <= sStr.length()-1)
//该函数能够将字符串 sStr 中的'<' 转换为"&lt;";
if (sStr.charAt(i) == '<')
sTmp1 = sTmp1.append("&lt;");
//也能够将字符串 sStr 中的'>' 转换为"&gt;";
} else if (sStr.charAt(i) == '>')
sTmp1 = sTmp1.append("&gt;");
}else
{
sTmp1 = sTmp1.append(sStr.substring(i,i+1));
}
i++;
}
String S2;
S2=sTmp1.toString();
return S2;
}
%>

<%!
String topic_id,title,author,content,aemail;
%>
<%
topic_id=request.getParameter("topicID");
title=returnToBr(returnToHTML(request.getParameter("title")));
author=returnToBr(returnToHTML(request.getParameter("author")));
content=returnToBr(returnToHTML(request.getParameter("content")));
aemail=returnToBr(returnToHTML(request.getParameter("aemail")));
if(aemail.equals("")){
aemail="null";
}
//转换中文
title=getStr(title);
author=getStr(author);
content=getStr(content);
aemail=getStr(aemail);
%>

```



```
<%  
String sqlinsert="insert into  reply(topicID,title,author,email,content) ";  
sqlinsert= sqlinsert+" Values('"+topic_id+"','"+title+"','"+author+"','"+aemail+"','"+content+"')";  
// 执行数据库操作  
workM.executeQuery(sqlinsert);  
%>  
<%@ include file="../book_store/head.inc"%>  
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">  
  <TBODY>  
    <tr><td align="left" height=25>  
      <% //显示用户名  
      if(session.getAttribute("username")!=null){  
        out.println(session.getAttribute("username"));  
      }  
      %> 当前位置: <a href="../index.jsp">首页</a> -&gt; 成员服务 </td>  
      <td align="right"><!--显示时间--> <%@ include file="date.inc"%> </td>  
    </tr>  
    .....部分代码省略（详细代码请参看附带光盘）  
  </tr>  
</table>  
<!--包含的尾文件-->  
<%@ include file="../member/footer.inc"%>
```

## 6. 修改帖子效果图及其源代码

单击修改链接, 则可以对对应的帖子进行修改。如图 21-6 所示。

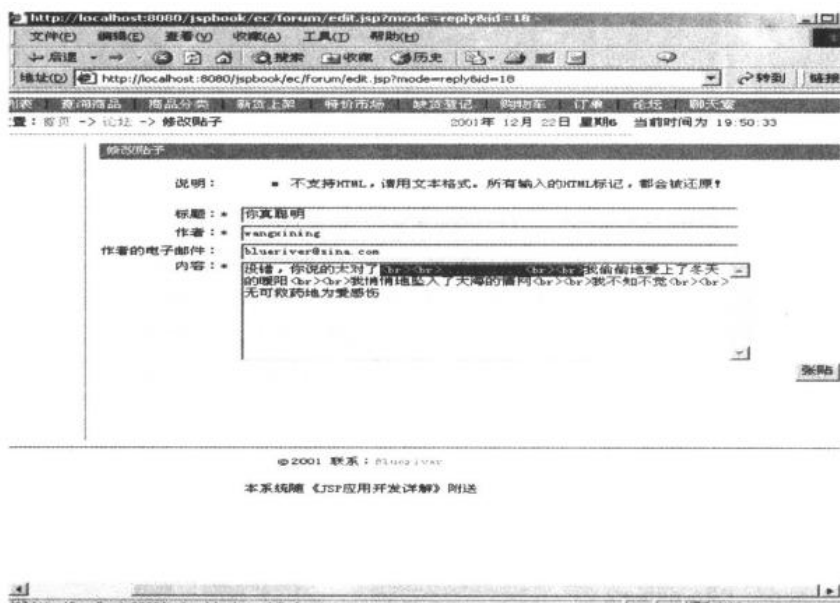


图 21-6 修改帖子效果图

文件 edit.jsp 的源代码如例程 21-5 所示。

例程 21-5

```

//头文件 header.inc
<%@ include file="../book_store/head.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<%@ page language="java" import="java.sql.*" %>
//调用用来执行数据库操作的 JavaBean (test.faq)
<jsp:useBean id="workM" scope="page" class="test.faq" />
//处理中文问题的自定义函数
<%!
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("GBK");
        String temp=new String(temp_t,"ISO8859_1");
        return temp;
    }
    catch(Exception e)
    {
    }
    return "null";
}
%>
<% //判断用户是否已经成功登录;
    //如果没有登录成功, 页面则会导向登录界面。
if(session.getAttribute("username")==null){
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
if(session.getAttribute("username")!=null){
    out.println(session.getAttribute("username"));
}
%> 当前位置: <a href="../index.jsp">首页</a> -&gt; 成员服务 </td>
            <td align="right"><!--显示时间--> <%@ include file="date.inc"%> </td>
        </tr>
        .....部分代码省略 (详细代码请参看附带光盘)
        <%
            String id,mode;
            id=request.getParameter("id");
            mode=request.getParameter("mode");

```

```

        if(mode.equals("topic"))
        {
            mode="topic";
        }
        else
        {
            mode="reply";
        }
        String strSQL="SELECT * FROM "+mode+" where id="+id;
        ResultSet RSa = workM.executeQuery(strSQL);
        while (RSa.next()) {
            String a_author,title,aemail,content;
            a_author=getStr(RSa.getString("author"));
            title=getStr(RSa.getString("title"));
            aemail=getStr(RSa.getString("email"));
            content=getStr(RSa.getString("content"));
        }
    %>

    <tr>
        <td align="right">标题: </td>
        <td width="20">*</td>
        <td width="500">
            <input type="hidden" name="id" value="<%=id%>">
            <input type="hidden" name="mode" value="<%=mode%>">
            <input type="text" name="title" size="64" value="<%=title%>">
        </td>
    </tr>
    <tr>
        <td align="right">作者: </td>
        <td width="20">*</td>
        <td width="500">
            <input type="text" name="author" size="64" value="<%=sess-
ion.getValue("username")%>" readonly>
        </td>
    </tr>
    <tr>
        <td align="right">作者的电子邮件: </td>
        <td width="20">&nbsp;</td>
        <td width="500">
            <input type="text" name="aemail" size="64" value="<%=
aemail%>">
        </td>
    </tr>
    <tr>
        <td align="right" valign="top">内容: </td>
        <td width="20" valign="top">*</td>
        <td>

```

```

        <textarea name="content" cols="64" rows="8">
<%=content%></textarea>
        </td>
    </tr>
    <tr>
        <td align="right">&nbsp;   </td>
        <td width="20">&nbsp;   </td>
        <td align="right">
            <input type="button" name="post" value=" 张 贴 "
onclick="sub()">
        </td>
    </tr>
</table>
</form>

<%
    }
    RSa.close();
%>
    .....部分代码省略（详细代码请参看附带光盘）
<!--尾文件-->
<%@ include file="./member/footer.inc"%>

```

## 7. 修改帖子逻辑处理页面及其源代码

单击修改链接，则可以对对应的帖子进行修改。修改帖子的逻辑处理在 edit\_ok.jsp 中进行。

文件 edit\_ok.jsp 的源代码如例程 21-6 所示。

例程 21-6

```

//调用用来执行数据库操作的 JavaBean
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="workM" scope="page" class="test.faq" />
.....部分代码省略（详细代码请参看附带光盘）
<%!
    String title,author,content,aemail;
%>
<%
    title=returnToBr(returnToHTML(request.getParameter("title")));
    author=returnToBr(returnToHTML(request.getParameter("author")));
    content=returnToBr(returnToHTML(request.getParameter("content")));
    aemail=returnToBr(returnToHTML(request.getParameter("aemail")));
    if(aemail.equals("")){
        aemail="null";
    }

    title=getStr(title);
    author=getStr(author);

```

```

content=getStr(content);
aemail=getStr(aemail);

%>
<%

    String mode,id;
    id=request.getParameter("id");
    mode=request.getParameter("mode");
    if(mode.equals("topic"))
    {
        mode="topic";
    }
    else
    {
        mode="reply";
    }

    String sqlinsert="update "+mode+" set title='"+title+"',author='"+author;
    sqlinsert = sqlinsert + "',email='"+aemail+"',content='"+content+" where id='"+id;
    workM.executeQuery(sqlinsert);
%>
//头文件 header.inc
<%@ include file="../book_store/head.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
            if(session.getAttribute("username")!=null){
            out.println(session.getAttribute("username"));
            }
            %> 当前位置: <a href="../index.jsp">首页</a> -&gt; <a href="index.jsp">论坛</a> -&gt; 修改贴
子 </td>
            <%@ include file="../member/date.inc"%>
            .....部分代码省略 (详细代码请参看附带光盘)
            <form name="form1" method="post" action="post.jsp">
                <table width="100%" border="0" cellspacing="0" cellpadding="0">
                    <tr><td colspan="3" height="20">您已经成功修改帖子 , <a
href="index.jsp">返回</a>! </td></tr>
                </table>
            </form>
        </td>
    </tr>
</table>
</td>
</tr>

```

```

</table>
<!--包含的尾文件-->
<%@ include file="../member/footer.inc"%>

```

## 8. 删除帖子的逻辑处理页面及其源代码

单击删除链接，则可以对对应的帖子进行删除。其源代码如例程 21-7 所示。

例程 21-7

```

<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="workM" scope="page" class="test.faq" />

<%
    String mode,id;
    id=request.getParameter("id");
    mode=request.getParameter("mode");
    if(mode.equals("topic"))
    {
        mode="topic";
    }
    else
    {
        mode="reply";
    }
    String sql="delete from "+mode+" where id="+id;
    workM.executeQuery(sql);
    response.sendRedirect("index.jsp");
%>

```

## 9. 发表新的主题页面及其源代码

发布新的主题帖子，如图 21-7 所示是通过页面 post.jsp 实现的。

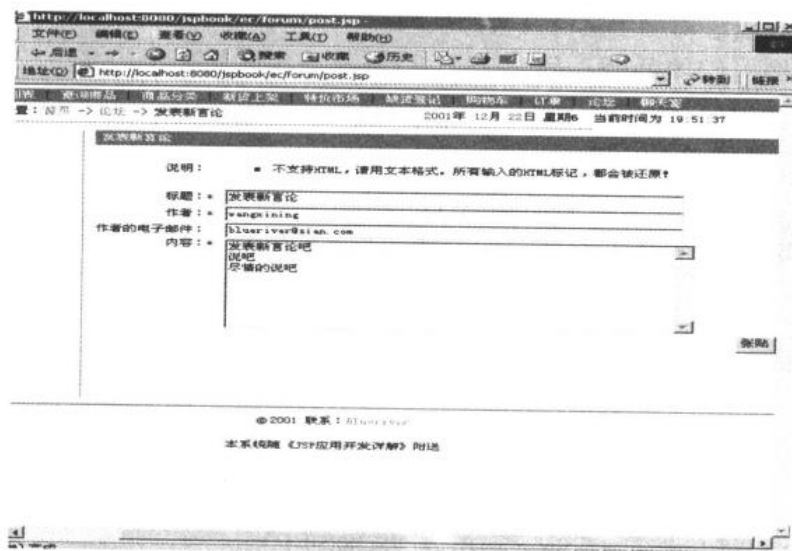


图 21-7 发布新主题页面

文件 post.jsp 源代码的如例程 21-8 所示。

例程 21-8

```
//头文件 header.inc
<%@ include file="../book_store/head.inc"%>
<% //判断用户是否已经登录；如果还没有登录，则页面会直接导向登录界面；
    //在实际的开发中，还应该判断登录的用户是否具有执行发布新的商品的权限；
    //也就是说，只有具有相应的权限的用户，才可以进行下面的操作；
if(session.getAttribute("username")==null){
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
%>
//调用用来显示时间的 JavaBean （dates.JspCalendar）
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type='dates.JspCalendar' />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
    <TBODY>
        <tr><td align="left" height=25>
            <% //显示用户名
if(session.getAttribute("username")!=null){
    out.println(session.getAttribute("username"));
}
%> 当前位置： <a href="../index.jsp">首页</a> -&gt; <a href="index.jsp">论坛</a> -&gt; 发表新
言论
            </td>
            <%@ include file="../member/date.inc"%>
        </tr>
        .....部分代码省略（详细代码请参看附带光盘）
    <!--尾文件-->
    <%@ include file="../member/footer.inc"%>
</table>
```

## 10. 发表新主题逻辑处理页面及其源代码

其源代码如例程 21-9 所示。

例程 21-9

```
//调用用来执行数据库操作的 JavaBean （test.faq）
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="workM" scope="page" class="test.faq" />
<%!
//处理中文问题的自定义函数
public String getStr(String str)
{
    try
    {
        String temp_p=str;
        byte[] temp_t=temp_p.getBytes("ISO8859-1");
        String temp=new String(temp_t);
        return temp;
    }
    catch (Exception e)
    {
        return str;
    }
}
```



```

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "null";
}
%>

```

<%! //处理输入的信息内容

//该函数能够将字符串 sStr 中的'\n'、'\r'转换为<br>;

//也能够将字符串 sStr 中的' '转换为&nbsp;;

```
public static String returnToBr(String sStr)
```

//如果 sStr 为 null 或者 sStr 为空值;

//则该函数会返回不发生变化的 sStr;

```
if (sStr == null || sStr.equals(""))
```

```
return sStr;
```

```
}
```

//定义 StringBuffer 型的变量 sTmp

```
StringBuffer sTmp = new StringBuffer();
```

```
int i = 0;
```

```
while (i <= sStr.length()-1)
```

//将字符串 sStr 中的'\n'、'\r'转换为<br>;

```
if (sStr.charAt(i) == '\n' || sStr.charAt(i) == '\r')
```

```
sTmp = sTmp.append("<br>");
```

```
} else if (sStr.charAt(i) == ' ')
```

将字符串 sStr 中的' '转换为&nbsp;;

```
sTmp = sTmp.append("&nbsp;");
```

```
}else
```

```
sTmp = sTmp.append(sStr.substring(i,i+1));
```

```
}
```

```
i++;
```

```
}
```

```
String S1;
```

```
S1=sTmp.toString();
```

```
return S1;
```

```
}
```

%>

<%! //处理输入的信息内容

//该函数能够将字符串 sStr 中的'<' 转换为"&lt;";

//也能够将字符串 sStr 中的'>' 转换为"&gt;";

```
public static String returnToHTML(String sStr)
```

//如果 sStr 为 null 或者 sStr 为空值;

//则该函数会返回不发生变化的 sStr;

```
if (sStr == null || sStr.equals(""))
```

```
return sStr;
}
//定义 StringBuffer 型的变量 sTmp
StringBuffer sTmp1 = new StringBuffer();
int i = 0;
while (i <= sStr.length()-1)
//该函数能够将字符串 sStr 中的'<' 转换为"&lt;";
if (sStr.charAt(i) == '<')
sTmp1 = sTmp1.append("&lt;");
//也能够将字符串 sStr 中的'>' 转换为"&gt;";
} else if (sStr.charAt(i) == '>')
sTmp1 = sTmp1.append("&gt;");
} else
{
sTmp1 = sTmp1.append(sStr.substring(i,i+1));
}
i++;
}
String S2;
S2=sTmp1.toString();
return S2;
}
%>

<%//定义变量
    String title,author,content,aemail;
%>
<% //应用函数 returnToBr 与 returnToHTML 对提交的信息进行处理
    title=returnToBr(returnToHTML(request.getParameter("title")));
    author=returnToBr(returnToHTML(request.getParameter("author")));
    content=returnToBr(returnToHTML(request.getParameter("content")));
    aemail=returnToBr(returnToHTML(request.getParameter("aemail")));
    if(aemail.equals("")){
        aemail="null";
    }
    //进行中文处理
    title=getStr(title);
    author=getStr(author);
    content=getStr(content);
    aemail=getStr(aemail);

%>
<% //执行数据库操作，发布新的帖子
    String sqlinsert="insert into    topic(title,author,email,content) ";
    sqlinsert= sqlinsert+"Values('"+title+"','"+author+"','"+aemail+"','"+content+"')";
    workM.executeQuery(sqlinsert);
```

```

%>
//头文件 header.inc
<%@ include file="../book_store/head.inc"%>
//调用用来显示时间的 JavaBean (dates.JspCalendar)
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />
<TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align="center">
  <TBODY>
    <tr><td align="left" height=25>
      <% //显示用户名
      if(session.getAttribute("username")!=null){
      out.println(session.getAttribute("username"));
      }
      %> 当前位置: <a href="../index.jsp">首页</a> -&gt; <a href="index.jsp">教程文章</a>
      -&gt; 发表言论成功 </td>
      <!--显示时间--><%@ include file="../member/date.inc"%>
    </tr>
    <TR bgColor=#3399ff>
      <TD height=1 colspan="2"><IMG height=1 src="images/spacer.gif"
      width=16></TD></TR>
    <tr><td height=10 colspan="2"><IMG height=1 src="images/spacer.gif"
    width=16></td></tr>
  </TBODY></TABLE>
  <table align="center" border="0" width="760" cellspacing="0" cellpadding="0" height="355">
    <tr>
      <td width="150" height="355" valign="top"></td>
      <td width="10" height="100%"></td>
      <td width="1" height="100%" bgcolor="#3399ff"></td>
      <td width="10" height="100%"></td>
      <td width="589" height="331" valign="top" background="images/bg1.gif">
        <table border="0" width="100%" cellspacing="0" cellpadding="0" height="307">
          <tr>
            <td width="100%" height="20" bgcolor="#3399ff">&nbsp;<font color="#ffffff">发表
            言论成功</font>
          </td>
        </tr>
      </td>
    </tr>
    <tr>
      <td>
        <form name="form1" method="post" action="post.jsp">
          <table width="100%" border="0" cellspacing="0" cellpadding="0">
            <tr><td colspan="3" height="20">您已经发表言论成功, <a
            href="index.jsp">返回</a>! </td></tr>
          </table>
        </form>
      </td>
    </tr>
  </table>

```

```
</td>
</tr>
</table>
<!--尾文件-->
<%@ include file="../member/footer.inc"%>
```

上述的论坛基本上是采用了 JSP+JavaBeans 模式实现的。

下面讲述一个完全用 Servlet 实现的论坛，读者可以通过阅读这两种不同模式的实现方式代码，来比较模式一与模式二两者之间的优缺点。

这个论坛使用的数据库与上述论坛的数据库为同一个。详细介绍请参看上述章节。

下面通过使用 Servlet 来实现论坛主帖子的列表，读者可以与前面讲述的 JSP 文件进行比较，体会两者之间的差异。

实现论坛主帖子的列表（如图 21-8 所示）是通过 servlet 文件 ec.forum\_index 实现的，详细的源代码可以阅读附带光盘中的 ec/forum\_index.java 文件。



图 21-8 论坛主帖子的列表(ec.forum\_index 文件)

ec.forum\_index.java 源代码如例程 21-10 所示。

例程 21-10

```
//包 ec
package ec;
//导入相关的类文件
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
import java.util.*;
import java.sql.*;
//定义的 public 类名称应该与文件名完全相同
public class forum_index extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**进行全局初始化*/
    public void init() throws ServletException {
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    //定义 public 型的函数 executeQuery, 返回值类型为 ResultSet
    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:faq";
        Connection connect = null;
        ResultSet rs = null;
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    //处理中文问题
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
            return temp;
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        return "null";
    }
}
```

```
}
/**处理 HTTP Get 请求*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    HttpSession session=request.getSession(true);
    out.println("<html>");
    out.println("<head><title>forum_index</title></head>");
    out.println("<body>");
    .....部分代码省略（详细代码请参看附带光盘）
    //算出共多少页
    int t;
    int mtotal;
    t=0;
    String strSQLsize="SELECT id FROM topic";
    ResultSet RSize = executeQuery(strSQLsize);
    try
    {
        while(RSize.next()){
            t=t+1;
        }
        //如果纪录总数除以每页的显示个数,余数大于 0,那么
        //逻辑页数应该为商+1;
        //否则, 逻辑页数应为所得的商
        //t 表示记录的总数
        //10 表示了每页所显示的个数
        //mtotal 为逻辑页数

        if((t%10)>0){
            mttotal=t/10+1;
        }else mttotal=t/10;
    }
    //定义变量
    String pageNo, mTmp;
    int i, j, k;
    //pageNo 表示请求的是第几页
    pageNo = request.getParameter("pageNo");
    //如果 pageNo 为 null 值, 则赋值为 1
    if(pageNo == null){
        pageNo = "1";
    }
    //j 表示的是 pageNo 对应的整型值
    j = Integer.parseInt(pageNo);
    //当 j 的值小于 1 时, 赋值为 1
    if(j < 1)
        j = 1;
```

```

//当j 的值大于总页数时，赋值为 mttotal
if(j > mttotal)
    j = mttotal;
// 读取数据库记录
String strSQL="SELECT * FROM topic order by id desc";
ResultSet RSa = executeQuery(strSQL);
//记录集移动到相应的位置
//j 为逻辑页数
for(k = 0;k < (j-1)*10;k++)
{
    RSa.next();
}
i = 0;
k = 1;
while (RSa.next()) {
    i = i + 1;
    //超过 10 条，循环退出
    if(i == 11)
    {
        k = 0;
        break;
    }
    out.print("<tr height='23'><td><li><a href='");
out.print("ec.forum_article?id="+RSa.getInt("id"));
    out.print(">"+(RSa.getString("title"))+"</a></td><td>");
out.print((RSa.getString("author"))+"</td><td>"+RSa.getDate("date")+"</td></tr>");
}
    i = i - k;
    RSa.close();
//以下代码行是用来显示页面数的
//当j 大于 1 时，就会显示“第一页”、“上一页”字样
if(j > 1)
{
    out.println("<a href='\"ec.forum_index?pageNo=1\"'>第一页</a>");
    int ii = Integer.parseInt(pageNo,10);
    if(ii > 1)
        ii = ii - 1;
    String ssTmp = Integer.toString(ii);
    out.println("<a href='\"ec.forum_index?pageNo="+ssTmp+"\"'>上一頁</a>");
}
//同样地，当j 小于总逻辑页数时，就会显示“下一页”、“最后页”字样
if(j < mttotal)
{
    int ii = Integer.parseInt(pageNo,10);
    if(ii < mttotal)
        ii = ii + 1;

```



```
String ssTmp = Integer.toString(ii);
out.println("<a href=\"ec.forum_index?pageNo="+ssTmp+"\">下一页</a>");
out.println("<a href=\"ec.forum_index?pageNo="+mtotal+"\">最后页</a>");
}
if(mtotal < j)
    j = mtotal;
    out.println("结果共"+mtotal+"页,显示第"+j+"页");
}
catch(Exception sqlEx)
{
    System.err.println(sqlEx.getMessage());
}
.....部分代码省略（详细代码请参看附带光盘）
}
/**释放资源*/
public void destroy() {
}
}
```

其他的 Servlet 文件的源代码在此就不再讲述，读者可以参考附带光盘中的源文件代码。

例如：查看每个帖子的内容 [http://localhost:8080/jspbook/servlet/ec.forum\\_article?id=137](http://localhost:8080/jspbook/servlet/ec.forum_article?id=137)，如图 21-9 所示。



图 21-9 查看帖子页面

读者可以通过这种以两者不同的方式实现同一功能的例子，以便能够比较在 JSP 开发中模式一与模式二两者之间的区别。

# 第 22 章 聊 天 室

本章讲述了聊天室的构建技术，我们分两节讨论。第一节介绍以 JSP+JavaBeans 技术来构建聊天室。第二节则用 Servlet 技术来实现同样的功能。

## 22.1 用 JSP+JavaBeans 实现的聊天室

### 1. 初始化聊天室

在开始使用聊天室之前，需要先初始化一些变量。这些变量用来保存聊天的用户列表、聊天的信息、聊天者的信息，如登录 IP 以及登录时间等。初始化的程序如下：

init.jsp 源代码如例程 22-1 所示。

例程 22-1

```
<%@ page import="java.util.Vector" %>
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.Hashtable" %>
<%@ page import="java.util.Date " %>
<html>
<head>
<title>初始化聊天室</title>
</head>
<body>
<%
//初始化聊天室信息
Vector chatMsgVector=new Vector();
chatMsgVector.addElement("<font color='red' size='2'>欢迎来到 blueriver 聊天室!!! <br>");
chatMsgVector.addElement("为中国今年的胜利加油</font><br>");
//初始化欢迎信息
getServletContext().setAttribute("chatMsg",chatMsgVector);
//初始化聊天室用户列表（blueriver、wangxining、蓝河之波）
Vector chatUser=new Vector();
chatUser.addElement("blueriver[管理员]");
chatUser.addElement("wangxining[管理员]");
chatUser.addElement("蓝河之波[管理员]");
getServletContext().setAttribute("chatUser",chatUser);
//初始化聊天主题
getServletContext().setAttribute("chatTopic","为中国今年的胜利加油");
%>
<%
Hashtable userLife=new Hashtable();
```

```

Hashtable userIP=new Hashtable();
Date initTime=new Date();
String initIP="192.168.1.100";
userLife.put("init",new Long(initTime.getTime()));
userIP.put("init",initIP);
getServletContext().setAttribute("userLife",userLife);
getServletContext().setAttribute("userIP",userIP);
%>
</body>
</html>

```

## 2. 初始化聊天信息状态

发送信息的初始化, 调用一次之后, 就会指向 sendmsg.jsp。  
sendmsg\_init.jsp 源代码如例程 22-2 所示。

例程 22-2

```

<html>
<title>发送聊天信息</title>
<head>
<script language="JavaScript" type="text/javascript">
function  setCookie()
{
    document.cookie=form1.toTalk.selectedIndex+"#";
}
</script>
</head>

<%@ page import="java.util.Vector" %>
<%@ page contentType="text/html;charset=gb2312" %>

<body bgcolor="#CCCC99">
<!--当用户第一次说话后设置 cookie 并将本帧定位到 sendmsg.jsp-->
<form action="sendmsg.jsp" method="post" onsubmit="setCookie()" id="form1">
    <table border="0" cellspacing="3" cellpadding="0">
        <tr>
            <td><%=session.getValue("username")%></td>
            <td align="left" colspan="2">
                <input name="message" style="HEIGHT: 21px; WIDTH: 367px">
            </td>
        </tr>
        <tr>
            <td align="center">
                对象
            </td>
            <td align="left">
                <!--在一个下拉列表中显示所有用户名供用户选择-->

```

```

<select name="toTalk">
  <option value="all">全体在聊</option>
  <%
    Vector tempuser=(Vector)getContext().getAttribute("chatUser");
    for (int i=0;i<tempuser.size();i++)
    {
      out.println("<option>");
      out.println(tempuser.elementAt(tempuser.size()-i-1).toString());
      out.println("</option>");
    }
  %>
</select>
<input type="submit" value="说话呀" name="submit">
</td>
<td>
  <a href="leavechat.jsp" target="_top">离开聊天室</a>
</td>
</tr>
</table>
</form>
</body>
</html>

```

### 3. 聊天室的主界面

用户成功登录系统之后，进入主页面 index.jsp，如图 22-1 所示。

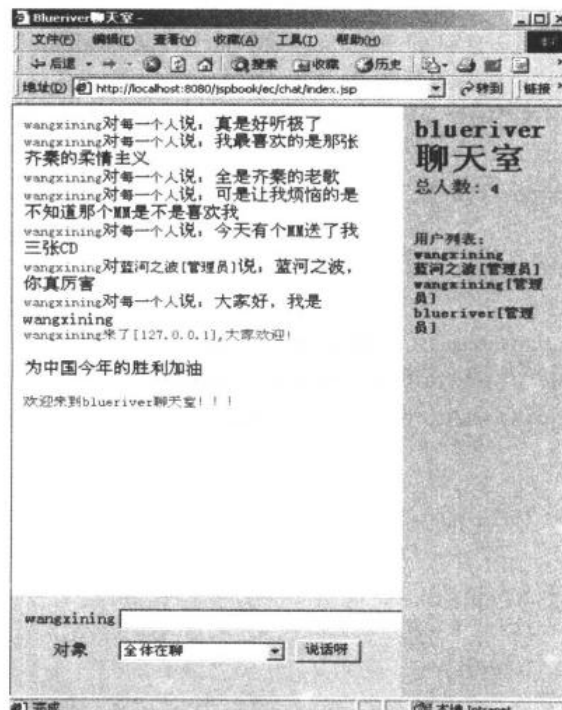


图 22-1 聊天室主界面

index.jsp 源代码如例程 22-3 所示。

例程 22-3

```

<% @ page import="java.util.Vector" %>
<% @ page contentType="text/html; charset=gb2312" %>
<% @ page import="java.util.Hashtable" %>
<% @ page import="java.util.Date" %>
<%
if(session.getAttribute("username")==null){
    response.sendRedirect("../member/login.jsp?url="+request.getRequestURI());
}
else
{
    //如果用户登录成功则在用户向量中加入新的用户名
    String username;
    Vector temp=new Vector();
    username=(String)session.getAttribute("username");
    temp=(Vector)getContext().getAttribute("chatUser");
    temp.addElement(username);
    getContext().setAttribute("chatUser",temp);
    //添加欢迎信息到消息向量里
    Vector tempmsg=(Vector)getContext().getAttribute("chatMsg");
    tempmsg.addElement("<font      size='2'      color='red'>"+username+"      来      了
["+request.getRemoteAddr()+"],大家欢迎!</font><br>");
    getContext().setAttribute("chatMsg",tempmsg);
    //用一个哈希表来记录用户在聊天室呆的时间
    Hashtable userLife=(Hashtable)getContext().getAttribute("userLife");
    //用一个哈希表记录在聊用户的 IP 地址
    Hashtable userIP=(Hashtable)getContext().getAttribute("userIP");
    //取得用户登录时间
    Date userTime=new Date();
    //记录用户登录时间
    userLife.put(username,new Long(userTime.getTime()));
    //记录用户 IP 地址
    userIP.put(username,request.getRemoteAddr());
    getContext().setAttribute("userLife",userLife);
    getContext().setAttribute("userIP",userIP);
}
%>
<html>
<head>
<title>Blueriver 聊天室</title>
</head>

<frameset cols="70%,*" border="0" framespacing="0" frameborder="NO">
    <frameset  rows="83%,*">

```

```

        <frame name="" src="showmsg.jsp" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="0">
        <frame name="" src="sendmsg_init.jsp" marginwidth="10" marginheight="10" scrolling="no"
frameborder="0">
    </frameset>
    <frame name="" src="listuser.jsp" marginwidth="10" marginheight="10" scrolling="no"
frameborder="0">
    </frameset><noframes></noframes>
</html>

```

由上述代码可以看出，如果用户没有成功登录，是不能够进入该页面的。读者也可以看出 index.jsp 是一个框架结构的页面，由三部分构成。

该主界面的主要部分是 showmsg.jsp，该部分是用来显示聊天信息的页面，该页面会自动刷新。

该主界面的下半部分是 sendmsg\_init.jsp，该部分是初始化发送信息的程序，调用一次后，导向 sendmsg.jsp。

该主界面的右边部分是 listuser.jsp，该部分显示所有的在线用户。

#### 4. 发送聊天信息

该部分是整个聊天室的最主要的部分，用来发送聊天信息。

Sendmsg.jsp 源代码如例程 22-4 所示。

例程 22-4

```

<%@ page import="java.util.Vector" %>
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.Hashtable" %>
<%@ page import="java.util.Date" %>
<html>
<title>发送聊天信息</title>
<head>
<script language="JavaScript" type="text/javascript">
function setCookie()
{
    document.cookie=form1.toTalk.selectedIndex+"#";
}
function setFocus()
{
    form1.message.focus();
    var cookieValue=document.cookie;
    var formValue=cookieValue.split("#");
    form1.toTalk.selectedIndex=formValue[1];
    _formValue=formValue[0].split(";");
}
</script>
</head>
<body bgcolor="#CCCCFF" onload="setFocus()">

```

```

<%!
//define a useful function which will be used by the following code
public void clearVector()
{
    Vector _tempchatMsg=(Vector)getContext().getAttribute("chatMsg");

    if (_tempchatMsg.size()>40)
    {
        _tempchatMsg.removeAllElements();
        _tempchatMsg.addElement("<font color='orange' size='2'>"+ "系统管理员清除了消息队
列，大家慢聊"+ "</font><br>");
    }
}
%>
<%
//*****
//初始化一些常用的变量
boolean boolcanupdatemsg=true;
Vector tempchatMsg=(Vector)getContext().getAttribute("chatMsg");
String cUserName=(String)session.getAttribute("username");
Date userTime=new Date();
String
timestamp="["+userTime.getHours()+":"+userTime.getMinutes()+":"+userTime.getSeconds()+"]";
//*****
%>
<%
//*****
//receive some parameter and analysis them
byte[] temp_t;
String temp_p;
temp_p=request.getParameter("message");
temp_t=temp_p.getBytes("ISO8859-1");
String temp=new String(temp_t);
byte[] talkTo_t;
String talkTo_p;
talkTo_p=request.getParameter("toTalk");
talkTo_t=talkTo_p.getBytes("ISO8859-1");
String talkTo=new String(talkTo_t);
%>
<%
//analysis the parameter
if(temp.startsWith("<"))
{
    temp="<font color='red' size='2'>请不要使用 html 标记"+timestamp+"</font><br>";
    tempchatMsg.addElement(temp);
    boolcanupdatemsg=false;
}

```



```

        getServletContext().setAttribute("chatMsg",tempchatMsg);
    }
    if (temp.endsWith("/>"))
    {
        temp="<font color='red' size='2'>请不要使用 html 标记"+timestamp+"</font><br>";
        tempchatMsg.addElement(temp);
        boolcanupdatemsg=false;
        getServletContext().setAttribute("chatMsg",tempchatMsg);
    }
    if (talkTo.equals("all"))
    {
        talkTo="每一个人";
    }
    %>
    <%
    cUserName="<font color='red' size='2'>"+cUserName+"</font>";
    talkTo="<font color='#0000FF' size='2'>"+talkTo+"</font>";

    if (boolcanupdatemsg==true)
    {
        //update some msg so that the user have the asbility to update the msg
        Hashtable userLife=(Hashtable)getServletContext().getAttribute("userLife");
        userLife.put((String)session.getValue("username"),new Long(userTime.getTime()));
        getServletContext().setAttribute("userLife",userLife);
        //*****
    }
    if (boolcanupdatemsg==true)
    {
        tempchatMsg.addElement(cUserName+"对"+talkTo+"说: "+temp);
        getServletContext().setAttribute("chatMsg",tempchatMsg);
    }
    %>
    <form action="sendmsg.jsp" method="post" onsubmit="setCookie()" id="form1">
    <table border="0" cellspacing="3" cellpadding="0">
    <tr>
    <td><%=session.getValue("username")%></td>
    <td align="left" colspan="2">
    <input name="message" style="HEIGHT: 21px; WIDTH: 367px">
    </td>
    </tr>
    <tr>
    <td align="center">
    对象
    </td>

```

```

<td align="left">
    <!-- 在一个下拉列表中显示所有用户名供用户选择-->
    <select name="toTalk">
        <option value="all">全体在聊</option>
        <%
            Vector tempuser=(Vector)getContext().getAttribute("chatUser");
            for (int i=0;i<tempuser.size();i++)
            {
                out.println("<option>");
                out.println(tempuser.elementAt(tempuser.size()-i-1).toString());
                out.println("</option>");
            }
        %>
    </select>
    <input type="submit" value="说话呀" name="submit">
</td>
<td>
    <a href="leavechat.jsp" target="_top">离开聊天室</a>
</td>
</tr>
</table>
</form>
</body>
</html>

```

## 5. 显示聊天信息

该部分是用来显示聊天信息的页面，如图 22-2 所示。

```

redSun对每个人说：孤独的夜晚。。。。。。
redSun对每个人说：呜呜呜呜呜呜。。。。。。
redSun对每个人说：真多没有人理我吗
redSun对每个人说：爱无悔
redSun对每个人说：有多少爱可以重来
redSun对每个人说：中国加油，加油中国！！！！
redSun对每个人说：大家好
redSun对蓝河之波[管理员]说：你好，蓝河之波，见到你真高兴。。。
redSun来了 [127.0.0.1], 大家欢迎！

jasvasun对wangxining[管理员]说：asdffff
jasvasun对每个人说：sdfsf
jasvasun对每个人说：wtt
为中国今年的胜利加油

欢迎来到blueriver聊天室！！

```

图 22-2 聊天信息

Showmsg.jsp 源代码如例程 22-5 所示。

```
<html>
<meta http-equiv="refresh" content="10">
<head>
<title>聊天室聊天信息列表</title>
</head>
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util. Vector" %>
<body bgcolor="#FFFFCC" >
<%
String tempmsg;
String cUserName=(String)session.getAttribute("username");
Vector chatMsg=(Vector)getContext().getAttribute("chatMsg");
for (int i=0;i<chatMsg.size();i++)
{
    String s_UserName;
    String o_UserName;
    int firstPos;
    int lastPos;
    boolean ifwhisper=false;

    tempmsg=chatMsg.elementAt(chatMsg.size()-i-1).toString();
    if (tempmsg.startsWith("#"))
    {
        ifwhisper=true;
    }
    if (ifwhisper)
    {
        firstPos=1;
        lastPos=tempmsg.indexOf("#",firstPos);
        s_UserName=tempmsg.substring(firstPos,lastPos);

        firstPos=lastPos+1;
        lastPos=tempmsg.indexOf("#",firstPos);
        o_UserName=tempmsg.substring(firstPos,lastPos);

        firstPos=lastPos+1;
        tempmsg=tempmsg.substring(firstPos,tempmsg.length());
        if(o_UserName.equals(cUserName))
        {
            tempmsg="<img src='new.gif'>" +tempmsg;
            out.println(tempmsg);
        }
        if (s_UserName.equals(cUserName))
        {
            tempmsg="<img src='new.gif'>" +tempmsg;
```

```
        out.println(tempmsg);
    }
}
else
{
    out.println(tempmsg+"<br>");
}

}
%>
</body>
</html>
```

## 6. 在线用户列表

该部分是用来显示在线用户列表的页面，如图 22-3 所示。

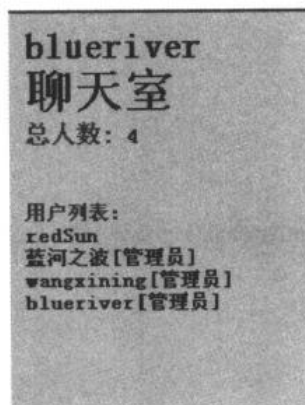


图 22-3 在线用户列表

Listuser.jsp 源代码如例程 22-6 所示。

例程 22-6

```
<%@ page import="java.util.Vector" %>
<%@ page contentType="text/html; charset=gb2312" %>
<html>
<head>
<title>聊天室用户列表</title>
</head>
<body bgcolor="#CCCC99">
<b><font size="+2" color="#000033">blueriver</font></b><br>
<b><font color="#000033" size="+3">聊天室</font></b><br>
<font color="#FF0033"><b>总人数:</b></font>
<font color="#0000FF" size='2'><b>
<%
Vector tempuser=(Vector)getContext().getAttribute("chatUser");
out.println(tempuser.size());
```

```

%>
</font></b><br><br><br>
<b> <font color="red" size="2"> 用户列表: </font><br>
<font size="2" color="#0000FF">
<%
for (int i=0;i<tempuser.size();i++)
{
    out.println(tempuser.elementAt(tempuser.size()-i-1).toString());
    out.println("<br>");
}
%>
</font></b>
<br>
</body>
</html>

```

## 7. 离开聊天室

该部分是表示用户离开聊天室，如图 22-4 所示。

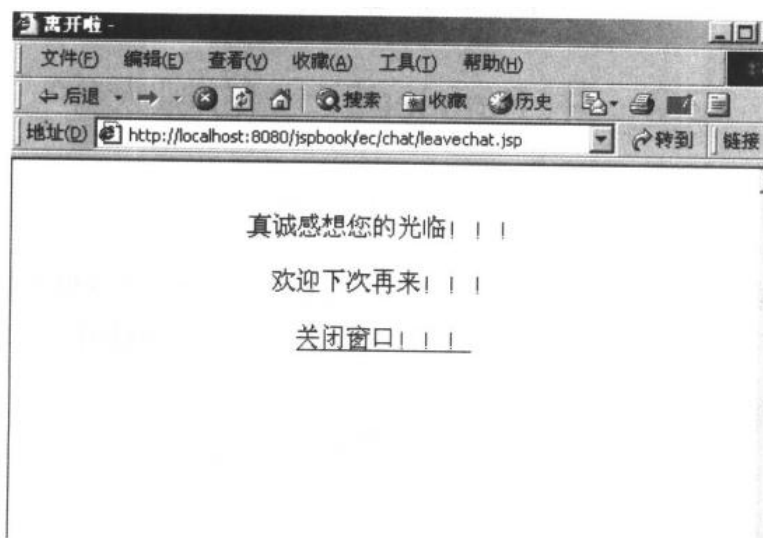


图 22-4 离开聊天室

Leavechat.jsp 源代码如例程 22-7 所示。

例程 22-7

```

<%@ page import="java.util.Vector" %>
<%@ page contentType="text/html; charset=gb2312" %>
<html>
<head>
<title>离开啦</title>
</head>
<body>
<%

```

```
String user=(String)session.getAttribute("username");
Vector temp=(Vector)getContext().getAttribute("chatUser");
temp.removeElement(user);
getContext().setAttribute("chatUser",temp);
%>
<%
temp=(Vector)getContext().getAttribute("chatMsg");
temp.addElement("<font color='#0000FF' size='2'>blueriver 管理员通告:");
temp.addElement(user+"离开了本聊天室,欢迎下次再来...<br></font>");
getContext().setAttribute("chatMsg",temp);
%>
<br>
<center>真诚感想您的光临!!! </center><br>
<center>欢迎下次再来!!! </center><br>
<center><a href="javascript:window.close();">关闭窗口!!! </a></center>
</body>
</html>
```

上述实现的聊天室是采用 JSP+JavaBeans 即模式一开发的。

## 22.2 用 Servlet 实现的聊天室

下面我们将讲述一下使用 Servlet 来实现同样功能的聊天室。

### 1. 初始化聊天室

在开始使用聊天室之前,需要先初始化一些变量。这些变量用来保存聊天的用户列表、聊天的信息、聊天者的信息,如登录 IP 以及登录时间等。初始化的程序如下。

chat\_init.java 源代码如例程 22-8 所示。

例程 22-8

```
//包 ec
package ec;
//导入相关的类文件
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
//定义的 public 类名称应该与文件名完全相同
public class chat_init extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
    }
    /**处理 HTTP 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
```

```

ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>初始化聊天室</title>");
    out.println("</head>");
    out.println("<body>");
    //初始化聊天室信息
    Vector chatMsgVector=new Vector();
    chatMsgVector.addElement("<font color='red' size='2'>欢迎来到 blueriver 聊天室!!! <br>");
    chatMsgVector.addElement("<font>为中国今年的胜利加油</font><br>");
    //初始化欢迎信息
    getServletContext().setAttribute("chatMsg",chatMsgVector);
    //初始化聊天室用户列表 (blueriver、wangxining、蓝河之波)
    Vector chatUser=new Vector();
    chatUser.addElement("blueriver[管理员]");
    chatUser.addElement("wangxining[管理员]");
    chatUser.addElement("蓝河之波[管理员]");
    getServletContext().setAttribute("chatUser",chatUser);
    //初始化聊天主题
    getServletContext().setAttribute("chatTopic","为中国今年的胜利加油");
    Hashtable userLife=new Hashtable();
    Hashtable userIP=new Hashtable();
    Date initTime=new Date();
    String initIP="192.168.1.100";
    userLife.put("init",new Long(initTime.getTime()));
    userIP.put("init",initIP);
    getServletContext().setAttribute("userLife",userLife);
    getServletContext().setAttribute("userIP",userIP);
    out.println("</body>");
    out.println("</html>");
}
/**释放资源*/
public void destroy() {
}
}

```

## 2. 初始化聊天信息状态

发送信息的初始化，调用一次之后，就会指向 chat\_sendmsg。  
chat\_sendmsg\_init.java 源代码如例程 22-9 所示。

例程 22-9

```

//包 ec
package ec;
//导入相关的类文件

```



```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
//定义的 public 类名称应该与文件名完全相同
public class chat_sendmsg_init extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
    }
    /**处理 HTTP 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        HttpSession session=request.getSession(true);
        out.println("<html>");
        out.println("<title>发送聊天信息</title>");
        out.println("<head>");
        out.println("<script language='JavaScript' type='text/javascript'>");
        out.println("function setCookie()");
        out.println("{");
        out.println("document.cookie=form1.toTalk.selectedIndex+'#';");
        out.println("}");
        out.println("</script>");
        out.println("</head>");
        out.println("<body bgcolor='\"#CCCC99\"'>");
        out.println("<form action='\"ec.chat_sendmsg\"' method='\"post\"' onsubmit='\"setCookie()\"'");
        id='\"form1\"'>");
        out.println("    <table border='\"0\"' cellspacing='\"3\"' cellpadding='\"0\"'>");
        out.println("        <tr>");
        out.println("            <td>"+session.getValue("username")+"</td>");
        out.println("            <td align='\"left\"' colspan='\"2\"'>");
        out.println("                <input name='\"message\"' style='\"HEIGHT: 21px; WIDTH:");
        367px\"'> ");
        out.println("            </td> ");
        out.println("        </tr> ");
        out.println("        <tr> ");
        out.println("            <td align='\"center\"'> ");
        out.println("            对象 ");
        out.println("        </td> ");
        out.println("        <td align='\"left\"'> ");
        out.println("            <select name='\"toTalk\"' ");
        out.println("                <option value='\"all\"'>全体在聊</option> ");
        Vector tempuser=(Vector)getContext().getAttribute("chatUser");
```

```

for (int i=0;i<tempuser.size();i++)
{
    out.println("<option>");
    out.println(tempuser.elementAt(tempuser.size()-i-1).toString());
    out.println("</option>");
}
out.println("        </select> ");
out.println("        <input type=\"submit\" value=\"说话呀\" name=\"submit\"> ");
out.println("    </td> ");
out.println("    <td> ");
out.println("        <a href=\"chat_leavechat\" target=\"_top\">离开聊天室</a> ");
out.println("    </td> ");
out.println(" </tr> ");
out.println(" </table> ");
out.println("</form> ");
out.println("</body> ");
out.println("</html> ");
}
/**释放资源*/
public void destroy() {
}
}

```

### 3. 聊天室的主界面

用户成功登录系统之后，进入主页面 chat\_index，如图 22-5 所示。

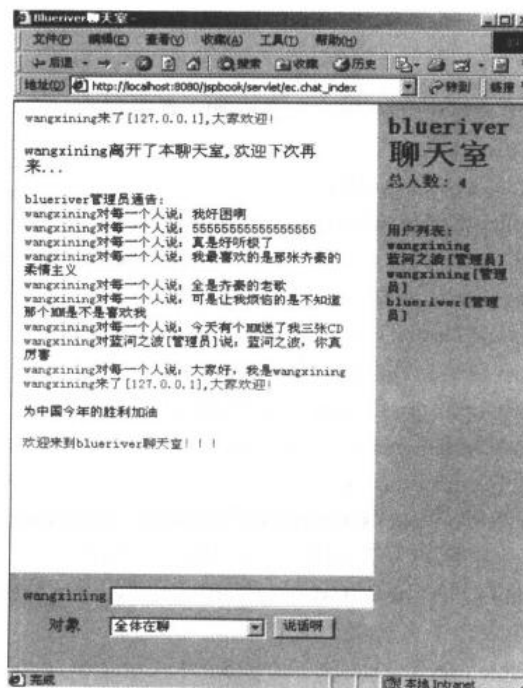


图 22-5 聊天室主界面（Servlet 文件）

chat\_index.java 源代码如例程 22-10 所示。

例程 22-10

```
//包 ec
package ec;
//导入相关的类文件
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
//定义的 public 类名称应该与文件名完全相同
public class chat_index extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
    }
    /**处理 HTTP 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        HttpSession session=request.getSession(true);
        if(session.getAttribute("username")==null){
            response.sendRedirect("ec.chat_login?url="+request.getRequestURI());
        }
        else
        {
            //如果用户登录成功则在用户向量中加入新的用户名
            String username;
            Vector temp=new Vector();
            username=(String)session.getAttribute("username");
            temp=(Vector)getServletContext().getAttribute("chatUser");
            temp.addElement(username);
            getServletContext().setAttribute("chatUser",temp);
            //添加欢迎信息到消息向量里
            Vector tempmsg=(Vector)getServletContext().getAttribute("chatMsg");
            tempmsg.addElement("<font size='2' color='red'>"+username+" 来了["+request.get-
RemoteAddr()+"],大家欢迎!</font><br>");
            getServletContext().setAttribute("chatMsg",tempmsg);
            //用一个哈希表来记录用户在聊天室呆的时间
            Hashtable userLife=(Hashtable)getServletContext().getAttribute("userLife");
            //用一个哈希表记录在聊用户的 IP 地址
            Hashtable userIP=(Hashtable)getServletContext().getAttribute("userIP");
            //取得用户登录时间
            Date userTime=new Date();
```

```

//记录用户登录时间
    userLife.put(username,new Long(userTime.getTime()));
//记录用户 IP 地址
    userIP.put(username,request.getRemoteAddr());
    getServletContext().setAttribute("userLife",userLife);
    getServletContext().setAttribute("userIP",userIP);
}
out.println("<html>");
out.println("<head>");
out.println("<title>Blueriver 聊天室</title>");
out.println("</head>");
out.println("<frameset cols=\"70%,*\" border=\"0\" framespacing=\"0\" frameborder=\"NO\">");
out.println("<frameset  rows=\"83%,*\">");
out.println("        <frame  name=\"\"  src=\"ec.chat_showmsg\"  marginwidth=\"10\"
marginheight=\"10\" scrolling=\"auto\" frameborder=\"0\">");
out.println("        <frame  name=\"\"  src=\"ec.chat_sendmsg_init\"  marginwidth=\"10\"
marginheight=\"10\" scrolling=\"no\" frameborder=\"0\">");
out.println("    </frameset>");
out.println("        <frame  name=\"\"  src=\"ec.chat_listuser\"  marginwidth=\"10\"
marginheight=\"10\" scrolling=\"no\" frameborder=\"0\">");
out.println("    </frameset><noframes></noframes>");
out.println("</html>");
}
/**释放资源*/
public void destroy() {
}
}

```

#### 4、用户登录

该部分是用来通过用户登录的页面，如图 22-6 所示。

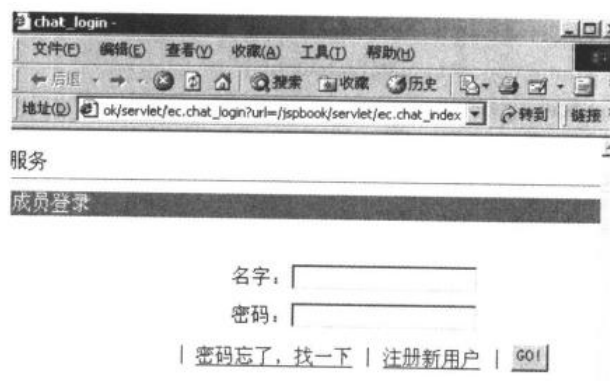


图 22-6 登录界面 (Servlet 文件)

chat\_login.java 源代码如例程 22-11 所示。

例程 22-11

```
//包 ec
package ec;
//导入相关的类文件
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
//定义的 public 类名称应该与文件名完全相同
public class chat_login extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
        String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println( e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql) {
        String sConnStr = "jdbc:odbc:faq";
        Connection connect = null;
        ResultSet rs = null;
        rs = null;
        try {
            connect = DriverManager.getConnection(sConnStr);
            Statement stmt = connect.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println(ex.getMessage());
        }
        return rs;
    }
    public String getStr(String str)
    {
        try
        {
            String temp_p=str;
            byte[] temp_t=temp_p.getBytes("ISO8859-1");
            String temp=new String(temp_t);
        }
    }
}
```

```

        return temp;
    }
    catch(Exception e)
    {

    }
    return "null";
}

/**处理 HTTP 请求*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    HttpSession session=request.getSession(true);
    out.println("<html>");
    out.println("<head><title>chat_login</title></head>");
    out.println("<body>");
    out.println(" <TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align=\"center\">");
    out.println("  <TBODY>");
    out.println("    <tr><td align=\"left\" height=25>");
    if(session.getAttribute("username")!=null)
    {
        out.println(session.getAttribute("username"));
    }
    out.println("  当前位置: <a href=\"../index.jsp\">首页</a> -&gt; 成员服务  </td>");
    out.println("    <td align=\"right\">  </td>");
    out.println("  </tr>");
    out.println("  <TR bgColor=#3399ff>");
    out.println("    <TD height=1 colspan=\"2\"><IMG height=1 src=\"images/spacer.gif\">");
    out.println("    width=16</TD></TR>");
    out.println("  <tr><td height=10 colspan=\"2\"><IMG height=1 src=\"images/spacer.gif\">");
    out.println("    width=16</td></tr>");
    out.println("  </TBODY></TABLE>");
    out.println("");
    out.println("    <table align=\"center\" border=\"0\" width=\"760\" cellspacing=\"0\"
cellpadding=\"0\" >");
    out.println("      height=\"355\">");
    out.println("    <tr>");
    out.println("      <td width=\"150\" height=\"355\" valign=\"top\">");
    out.println("    </td>");
    out.println("      <td width=\"10\" height=\"100%\"></td>");
    out.println("      <td width=\"1\" height=\"100%\" bgcolor=\"#3399ff\"></td>");
    out.println("      <td width=\"10\" height=\"100%\"></td>");
    out.println("      <td width=\"589\" height=\"331\" valign=\"top\"

```

```

background="\images/bg1.gif">");
    out.println("        <table border=\"0\" width=\"100%\" cellspacing=\"0\" cellpadding=\"0\"
height=\"307\">");
    out.println("                <tr>");
    out.println("                        <td width=\"100%\" colspan=\"2\" height=\"20\"
bgcolor=\"#3399ff\">&nbsp;  <font color=\"#ffffff\">成员登录</font>");
    out.println("                                </td>");
    out.println("                </tr>");
    out.println("");
    out.println("        <tr><td align=\"right\" height=\"32\" width=\"40%\">");
    out.println("        <font color=red></font>");
    out.println("        </td>");
    out.println("        </tr>");
    out.println("        <form action=\"\" method=\"post\">");
    out.println("        <tr><td align=\"right\" height=\"32\" width=\"40%\">名字: </td>");
    out.println("                <td>");
    out.println("                        <input type=\"hidden\" name=\"return\" value=\"\">");
    out.println("                        <input type=\"text\" name=\"logname\" value=\"\">");
    out.println("                </td>");
    out.println("        </tr>");
    out.println("        <tr>");
    out.println("                <td align=\"right\" height=\"32\">密码: </td>");
    out.println("                <td>");
    out.println("                        <input type=\"password\" name=\"logpass\">");
    out.println("                </td>");
    out.println("        </tr>");
    out.println("        <tr>");
    out.println("                <td align=\"center\" colspan=\"2\" height=\"32\">|");
    out.println("                <a href=\"../member/findpass.jsp\">密码忘了, 找一下</a> |");
    out.println("                <a href=\"../member/reg.jsp\">注册新用户</a> |");
    out.println("                <input type=\"submit\" name=\"login\" value=\"GO!\">");
    out.println("                </td>");
    out.println("        </tr>");
    out.println("        </form>");
    out.println("        <tr> ");
    out.println("                <td colspan=\"2\" align=\"right\">&nbsp;  </td>");
    out.println("");
    out.println("        </tr>");
    out.println("    ");
    out.println("    </table> ");
    out.println("    </td>");
    out.println("    </tr>");
    out.println("    </table>");
    out.println("</body></html>");
}

```



```
/**处理 HTTP 请求*/
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    HttpSession session=request.getSession(true);
    String logname,logpass;
    boolean loginAttempt = false;
    boolean loginOK = false;
    String errorMessage = "请您登录";

    if(request.getParameterValues("login") != null
    && request.getParameterValues("login")[0].trim().equals("GO!")
    &&request.getParameterValues("logname") != null
    &&request.getParameterValues("logpass") != null)
    {
        loginAttempt = true;
    }
    if (loginAttempt)
    {
        logname=request.getParameter("logname");
        logpass=request.getParameter("logpass");
        logname=getStr(logname);
        logpass=getStr(logpass);
        String sql="select * from member where logname='"+logname+"' and password='"+logpass+"'";
        ResultSet RS=executeQuery(sql);
        int rowcount=0;
        try
        {
            while(RS.next())
            {
                rowcount++;
            }
        }
        catch(Exception e)
        {

        }
        if(rowcount!=0)
        {
            errorMessage="成功登录";
            session.setAttribute("username",logname);
            loginOK=true;

            if(loginOK){
                String url;
```

```

        url=request.getParameter("url");
        out.println(url);
        if(url==null){
            response.sendRedirect("http://localhost:9090/test/ec");
        }
        else{
            response.sendRedirect("http://localhost:9090"+url);
        }
    }
    }else{
        errorMessage="您的用户名或者密码不正确";
        session.setAttribute("username","");
    }
}

out.println("<html>");
out.println("<head><title>chat_login</title></head>");
out.println("<body>");
out.println(" <TABLE border=0 cellPadding=0 cellSpacing=0 width=760 align=\"center\">");
out.println("  <TBODY>");
out.println("    <tr><td align=\"left\" height=25>");
if(session.getAttribute("username")!=null)
{
    out.println(session.getAttribute("username"));
}
out.println("  当前位置: <a href=\"../index.jsp\">首页</a> -&gt; 会员服务  </td>");
out.println("    <td align=\"right\">  </td>");
out.println("  </tr>");
out.println("  <TR bgColor=#3399ff>");
out.println("    <TD height=1 colspan=\"2\"><IMG height=1 src=\"images/spacer.gif\">");
out.println("  width=16></TD></TR>");
out.println("    <tr><td height=10 colspan=\"2\"><IMG height=1 src=\"images/spacer.gif\">");
out.println("  width=16></td></tr>");
out.println("  </TBODY></TABLE>");
out.println("");
out.println("    <table  align=\"center\"  border=\"0\"  width=\"760\"  cellspacing=\"0\"
cellpadding=\"0\" ");
out.println("height=\"355\">");
out.println("    <tr>");
out.println("      <td width=\"150\" height=\"355\" valign=\"top\">");
out.println("");
out.println("      </td>");
out.println("      <td width=\"10\" height=\"100%\"></td>");
out.println("      <td width=\"1\" height=\"100%\" bgcolor=\"#3399ff\"></td>");
out.println("      <td width=\"10\" height=\"100%\"></td>");
out.println("      <td          width=\"589\"          height=\"331\"          valign=\"top\"
background=\"images/bg1.gif\">");

```

```

        out.println("        <table border=\"0\" width=\"100%\" cellpadding=\"0\"
height=\"307\">");
        out.println("                <tr>");
        out.println("                        <td width=\"100%\" colspan=\"2\" height=\"20\"
bgcolor=\"#3399ff\">&nbsp;  <font color=\"#ffffff\">成员登录</font>");
        out.println("                                </td>");
        out.println("                </tr>");
        out.println("");
        out.println("        <tr><td align=\"right\" height=\"32\" width=\"40%\">");
        out.println("        <font color=red>+errorMessage+</font>");
        out.println("        </td>");
        out.println("        </tr>");
        out.println("        <form action=\"\" method=\"post\">");
        out.println("        <tr><td align=\"right\" height=\"32\" width=\"40%\">名字: </td>");
        out.println("                <td>");
        out.println("                        <input type=\"hidden\" name=\"return\" value=\"\">");
        out.println("                        <input type=\"text\" name=\"logname\" value=\"\">");
        out.println("                </td>");
        out.println("        </tr>");
        out.println("        <tr>");
        out.println("                <td align=\"right\" height=\"32\">密码: </td>");
        out.println("                <td>");
        out.println("                        <input type=\"password\" name=\"logpass\">");
        out.println("                </td>");
        out.println("        </tr>");
        out.println("        <tr>");
        out.println("                <td align=\"center\" colspan=\"2\" height=\"32\">|");
        out.println("                <a href=\"../member/findpass.jsp\">密码忘了, 找一下</a> |");
        out.println("                <a href=\"../member/reg.jsp\">注册新用户</a> |");
        out.println("                <input type=\"submit\" name=\"login\" value=\"GO!\">");
        out.println("                </td>");
        out.println("        </tr>");
        out.println("        </form>");
        out.println("        <tr> ");
        out.println("                <td colspan=\"2\" align=\"right\">&nbsp;  </td>");
        out.println("");
        out.println("        </tr>");
        out.println("        ");
        out.println("        </table> ");
        out.println("                </td>");
        out.println("        </tr>");
        out.println("        </table>");
        out.println("</body></html>");

}
/**释放资源*/

```

```
public void destroy() {  
    }  
}
```

## 5. 发送聊天信息

该部分是整个聊天室的最主要的部分，用来发送聊天信息。

chat\_sendmsg.java 源代码如例程 22-12 所示。

例程 22-12

```
//包 ec  
package ec;  
//导入相关的类文件  
import javax.servlet.*;  
import javax.servlet.http.*;  
import java.io.*;  
import java.util.*;  
import java.sql.*;  
//定义的 public 类名称应该与文件名完全相同  
public class chat_sendmsg extends HttpServlet {  
    private static final String CONTENT_TYPE = "text/html; charset=GBK";  
    /**初始化变量*/  
    public void init() throws ServletException {  
    }  
    //define a useful function which will be used by the following code  
    public void clearVector()  
    {  
        Vector _tempchatMsg=(Vector)getContext().getAttribute("chatMsg");  
  
        if (_tempchatMsg.size()>40)  
        {  
            _tempchatMsg.removeAllElements();  
            _tempchatMsg.addElement("<font color='orange' size='2'>"+ "系统管理员  
清除了消息队列，大家慢聊"+ "</font><br>");  
        }  
    }  
  
    /**处理 HTTP 请求*/  
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {  
        response.setContentType(CONTENT_TYPE);  
        PrintWriter out = response.getWriter();  
        out.println("<html>");  
        out.println("<head><title>chat_sendmsg</title></head>");  
        out.println("<body>");  
        out.println("<p>The servlet has received a GET. This is the reply.</p>");  
        out.println("</body></html>");  
    }  
}
```

```

    }
    /**处理 HTTP 请求*/
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        HttpSession session=request.getSession(true);
        out.println("<html>");
        out.println("  <title>发送聊天信息</title>");
        out.println("<head>");
        out.println("<script language=\"JavaScript\" type=\"text/javascript\">");
        out.println("function  setCookie()");
        out.println("{");
        out.println("  document.cookie=form1.toTalk.selectedIndex+"\\"#\\";");
        out.println("}");
        out.println("function setFocus()");
        out.println("{");
        out.println("  form1.message.focus();");
        out.println("  var cookieValue=document.cookie;");
        out.println("  var formValue=cookieValue.split(\"#\");");
        out.println("  form1.toTalk.selectedIndex=formValue[1];");
        out.println("  _formValue=formValue[0].split(\";\");");
        out.println("}");
        out.println("</script>");
        out.println("</head>");
        out.println("<body bgcolor=\"\#CCCCFF\" onload=\"setFocus()\">");

        /*******
*
        //初始化一些常用的变量
        boolean boolcanupdatemsg=true;
        Vector tempchatMsg=(Vector)getContext().getAttribute("chatMsg");
        String cUserName=(String)session.getAttribute("username");
        Date userTime=new Date();
        String
timestamp=["+userTime.getHours()+":"+userTime.getMinutes()+":"+userTime.getSeconds()+"];
        /*******
*

        /*******
*

        //receive some parameter and analysis them
        byte[] temp_t;
        String temp_p;
        temp_p=request.getParameter("message");

```

```
temp_t=temp_p.getBytes("ISO8859-1");
String temp=new String(temp_t);

byte[] talkTo_t;
String talkTo_p;
talkTo_p=request.getParameter("toTalk");
talkTo_t=talkTo_p.getBytes("ISO8859-1");
String talkTo=new String(talkTo_t);

//analysis the parameter
if(temp.startsWith("<"))
{
    temp="<font color='red' size='2'>请不要使用 html 标记"+timestamp+"</font><br>";
    tempchatMsg.addElement(temp);
    boolcanupdatemsg=false;
    getServletContext().setAttribute("chatMsg",tempchatMsg);
}

if (temp.endsWith("/>"))
{
    temp="<font color='red' size='2'>请不要使用 html 标记"+timestamp+"</font><br>";
    tempchatMsg.addElement(temp);
    boolcanupdatemsg=false;
    getServletContext().setAttribute("chatMsg",tempchatMsg);
}

if (talkTo.equals("all"))
{
    talkTo="每一个人";
}

cUserName="<font color='red' size='2'>"+cUserName+"</font>";
talkTo="<font color='#0000FF' size='2'>"+talkTo+"</font>";

if (boolcanupdatemsg==true)
{
    //update some msg so that the user have the asbility to update the msg
    Hashtable userLife=(Hashtable)getServletContext().getAttribute("userLife");
    userLife.put((String)session.getValue("username"),new Long(userTime.getTime()));
    getServletContext().setAttribute("userLife",userLife);
    //*****
}
if (boolcanupdatemsg==true)
{
```

```

tempchatMsg.addElement(cUserName+"对"+talkTo+"说: "+temp);
getServletContext().setAttribute("chatMsg",tempchatMsg);

}

out.println("<form action=\"\" method=\"post\" onsubmit=\"setCookie()\" id=\"form1\">");
out.println("    <table border=\"0\" cellspacing=\"3\" cellpadding=\"0\">");
out.println("        <tr>");
out.println("            <td>"+session.getValue("username")+"</td>");
out.println("            <td align=\"left\" colspan=\"2\">");
out.println("                <input name=\"message\" style=\"HEIGHT: 21px; WIDTH: 367px\">");
out.println("            </td>");
out.println("        </tr>");
out.println("        <tr>");
out.println("            <td align=\"center\">");
out.println("                对象");
out.println("            </td>");
out.println("            <td align=\"left\">");
out.println("                <!--在一个下拉列表中显示所有用户名供用户选择-->");
out.println("                <select name=\"toTalk\">");
out.println("                    <option value=\"all\">全体在聊</option>");
out.println("                </select>");
out.println("                <input type=\"submit\" value=\"说话呀\" name=\"submit\">");
out.println("            </td>");
out.println("            <td>");
out.println("                <a href=\"ec.chat_leavechat\" target=\"_top\">离开聊天室</a>");
out.println("            </td>");
out.println("        </tr>");
out.println("    </table>");
out.println("</form>");
out.println("</body>");
out.println("</html>");
}

/**释放资源*/
public void destroy() {
}
}

```



## 6. 显示聊天信息

该部分是用来显示聊天信息的页面，如图 22-7 所示。

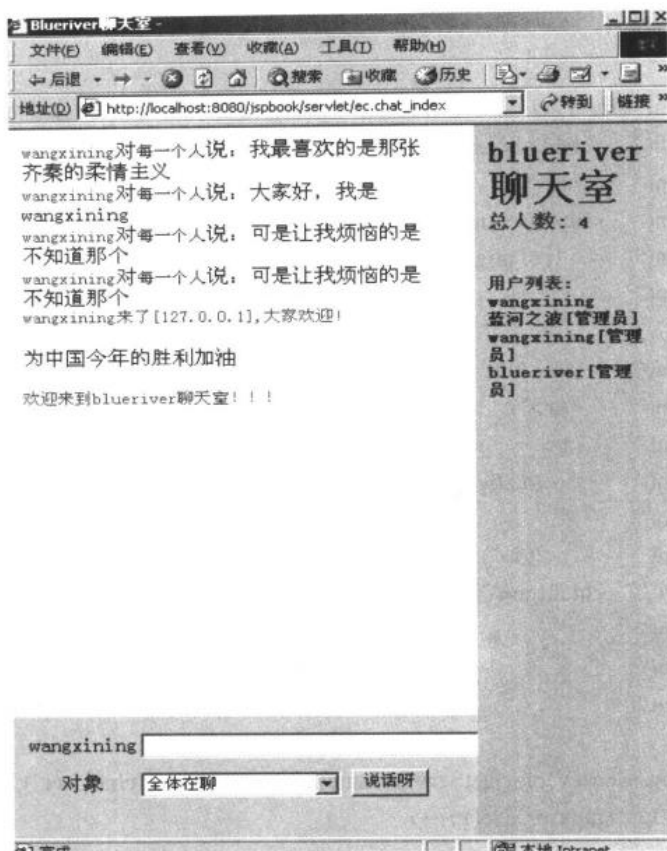


图 22-7 显示聊天信息(Servlet 文件)

chat\_showmsg.java 源代码如例程 22-13 所示。

例程 22-13

```
//包 ec
package ec;
//导入相关的类文件
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
//定义的 public 类名称应该与文件名完全相同
public class chat_showmsg extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
    }
    /**处理 HTTP 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
```

```

ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    HttpSession session=request.getSession(true);
    out.println("<html>");
    out.println("<meta http-equiv=\"refresh\" content=\"10\">");
    out.println("<head>");
    out.println("<title>聊天室聊天信息列表</title>");
    out.println("</head>");
    out.println("<body bgcolor=\"#FFFFCC\" >");

    String tempmsg;
    String cUserName=(String)session.getAttribute("username");
    Vector chatMsg=(Vector)getServletContext().getAttribute("chatMsg");
    for (int i=0;i<chatMsg.size();i++)
    {
        String s_UserName;
        String o_UserName;
        int firstPos;
        int lastPos;
        boolean ifwhisper=false;

        tempmsg=chatMsg.elementAt(chatMsg.size()-i-1).toString();
        if (tempmsg.startsWith("#"))
        {
            ifwhisper=true;
        }
        if (ifwhisper)
        {
            firstPos=1;
            lastPos=tempmsg.indexOf("#",firstPos);
            s_UserName=tempmsg.substring(firstPos,lastPos);

            firstPos=lastPos+1;
            lastPos=tempmsg.indexOf("#",firstPos);
            o_UserName=tempmsg.substring(firstPos,lastPos);

            firstPos=lastPos+1;
            tempmsg=tempmsg.substring(firstPos,tempmsg.length());
            if(o_UserName.equals(cUserName))
            {
                tempmsg="<img src='new.gif'>" +tempmsg;
                out.println(tempmsg);
            }
            if (s_UserName.equals(cUserName))
            {

```

```
        tempmsg="<img src='new.gif'>" + tempmsg;  
        out.println(tempmsg);  
    }  
    }  
    else  
    {  
        out.println(tempmsg + "<br>");  
    }  
}  
out.println("</body>");  
out.println("</html>");  
}  
/**释放资源*/  
public void destroy() {  
}  
}
```

## 7. 在线用户列表

该部分是用来显示在线用户列表的页面，如图 22-8 所示。



图 22-8 在线用户列表（Servlet 文件）

chat\_listuser.java 源代码如例程 22-14 所示。

例程 22-14

```
//包 ec  
package ec;  
//导入相关的类文件  
import javax.servlet.*;
```

```

import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
//定义的 public 类名称应该与文件名完全相同
public class chat_listuser extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
    }
    /**处理 HTTP 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("    <head>");
        out.println("<title>聊天室用户列表</title>");
        out.println("</head>");
        out.println("<body bgcolor=\"#CCCC99\">");
        out.println("<b><font size=\"+2\" color=\"#000033\">blueriver</font></b><br>");
        out.println("<b><font color=\"#000033\" size=\"+3\">聊天室</font></b><br>");
        out.println("<font color=\"#FF0033\"><b>总人数:</b></font>");
        out.println("<font color=\"#0000FF\" size=\"2\"><b> ");
        Vector tempuser=(Vector)getServletContext().getAttribute("chatUser");
        out.println(tempuser.size());
        out.println("</font></b><br><br><br>");
        out.println("<b> <font color=\"red\" size=\"2\"> 用户列表: </font><br>");
        out.println("<font size=\"2\" color=\"#0000FF\">");
        for (int i=0;i<tempuser.size();i++)
        {
            out.println(tempuser.elementAt(tempuser.size()-i-1).toString());
            out.println("<br>");
        }
        out.println("</font></b>");
        out.println("<br>");
        out.println("</body>");
        out.println("</html>");
    }
    /**释放资源*/
    public void destroy() {
    }
}

```

## 8. 离开聊天室

该部分是表示用户离开聊天室，如图 22-9 所示。

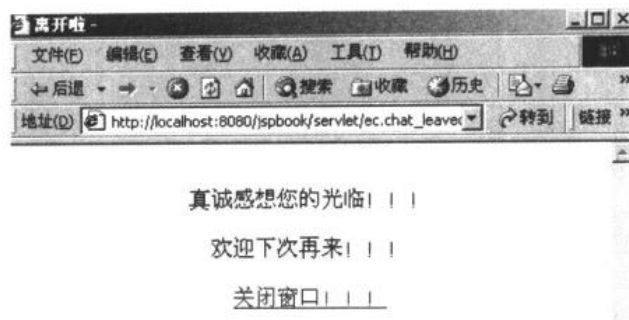


图 22-9 离开聊天室 (Servlet 文件)

chat\_leavechat.java 源代码如例程 22-15 所示。

例程 22-15

```
//包 ec
package ec;
//导入相关的类文件
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
//定义的 public 类名称应该与文件名完全相同
public class chat_leavechat extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GBK";
    /**初始化变量*/
    public void init() throws ServletException {
    }
    /**处理 HTTP 请求*/
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        HttpSession session=request.getSession(true);
        out.println("<html>");
        out.println("<head>");
        out.println("<title>离开啦</title>");
        out.println("</head>");
        out.println("<body>");
        String user=(String)session.getAttribute("username");
        Vector temp=(Vector)getServletContext().getAttribute("chatUser");
```

```
temp.removeElement(user);
getServletContext().setAttribute("chatUser",temp);
temp=(Vector)getServletContext().getAttribute("chatMsg");
temp.addElement("<font color='#0000FF' size='2'>blueriver 管理员通告:");
temp.addElement(user+"离开了本聊天室,欢迎下次再来...<br></font>");
getServletContext().setAttribute("chatMsg",temp);
out.println("<br>");
out.println("<center>真诚感想您的光临!!! </center><br>");
out.println("<center>欢迎下次再来!!! </center><br>");
out.println("<center><a href='\"javascript:window.close();\"'>关闭窗口!!! </a></center>");
out.println("</body>");
out.println("</html>");
}
/**释放资源*/
public void destroy() {
}
}
```

---

